12-1-1997

An Examination of the Strengths and Weaknesses of Newton's Method for Nonlinear Optimization

Renita Demalade
Western Kentucky University

AN EXAMINATION OF THE STRENGTHS AND WEAKNESSES OF NEWTON'S METHOD FOR NONLINEAR OPTIMIZATION

A Thesis

Presented to

the Faculty of the Department of Mathematics

Western Kentucky University

Bowling Green, Kentucky

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Renita Ponder DeMalade

December 1997

AN EXAMINATION OF THE STRENGTHS AND WEAKNESSES OF NEWTON'S METHOD FOR NONLINEAR OPTIMIZATION

Date Recommended 11/20/97

Mark P. Robinson

Director of Thesis

John S. Spraker

Daniel C. Bil

Edmon Gray 12/9/97

Dean, Graduate Studies and Research Date

ACKNOWLEDGEMENTS

In preparing and writing this thesis I have had the benefit of advice and help from many in the Mathematics department of Western Kentucky University but none so much as Dr. Mark Robinson who has gone above and beyond the requirements placed upon him in directing this thesis. His constant encouragement to look beyond the obvious and his attention to detail were the building blocks that enabled me not only to complete the task at hand but also to develop an appreciation for mathematical literature and the work involved in its research.

I would also like to acknowledge the other members of my committee: Dr. John Spraker and Dr. Daniel Biles for the time and insights offered me in completing this thesis. The extra effort, offered and arranged by long distance, shows the quality and leadership for which Western Kentucky University can proudly claim as its own.

TABLE OF CONTENTS

AN EXAMINATION OF THE STRENGTHS AND WEAKNESSES OF NEWTON'S METHOD FOR NONLINEAR OPTIMIZATION

Renita Ponder DeMalade December 1997 77 Pages

Directed by: Dr. Mark Robinson, Dr. John Spraker, and Dr. Daniel Biles

Department of Mathematics Western Kentucky University

This thesis begins with the history of operations research and introduces two of its major branches, linear and nonlinear optimization. While other methods are mentioned, the focus is on analytical methods used to solve nonlinear optimization problems. We briefly look at some of the most effective constrained methods for nonlinear optimization and then show how unconstrained methods often play a role in developing effective constrained optimization algorithms. In particular we examine Newton and steepest descent methods, focusing primarily on Newton/quasi-Newton methods. Because Newton's method is primarily viewed as a root-finding method, we start with the basic root-finding algorithm for single variable functions and show its progression into a useful, and often efficient, multivariable optimization algorithm. Comparisons are made between a pure Newton algorithm and a modified Newton algorithm as well as between a pure steepest descent algorithm and a modified steepest descent algorithm. In examining nonlinear functions of varying complexity, we note some of the considerations that must be made when choosing an optimization program as well as some of the difficulties that arise when using Newton's method or steepest descent methods for the optimization of a nonlinear function.

CHAPTER I

INTRODUCTION

Operations research can be described as the study of optimization. Its purpose is to use a scientific approach to decision making in order to determine how best to allocate scarce resources under given constraints. Although there are several successful strategies that can be employed to study optimization such as operational exercises, gaming and simulation, we will be focusing primarily on analytical methods.

The study of operations research has several branches, two of which are linear and nonlinear optimization. Linear programming offers a multipurpose algorithm, the simplex method, which is applicable for most applied linear problems although for extremely large problems an alternative algorithm exists that is considered more efficient. Nonlinear optimization, because of the vast number of ways a program can be considered nonlinear, offers no such multipurpose algorithm but much work has been done to identify special cases of nonlinear optimization for which suitable algorithms have been developed. Nonlinear programming can be separated into constrained and unconstrained optimization, both of which will be discussed in this thesis.

We will briefly discuss some of the most effective constrained methods and note that these methods often incorporate unconstrained optimization techniques such as the gradient search method (also known as steepest descent) and Newton/quasi-Newton methods. The two unconstrained methods that will

be considered, steepest descent and Newton/quasi-Newton, are line-search methods. Another category of unconstrained optimization methods involves trust regions but we will focus primarily on line-search methods and in particular the two previously mentioned methods.

The major focus of this thesis is to consider Newton methods and convergence properties possessed by these algorithms. We will be looking at such issues as

1) how close the initial trial solutions must be in order to guarantee quadratic convergence,

2) considerations in choosing stopping criteria for an algorithm,

3) how convergence of an algorithm is affected by the stopping criteria that are chosen,

4) comparing the paths taken by the various algorithms.

In Chapter Two of this thesis we introduce the necessary notation, definitions and preliminaries needed to follow the work that is to be presented.

In Chapter Three the historical significance of operations research will be discussed as well as an introduction to the general optimization problem. A brief discussion of linear programming methods and the simplex algorithm sets the stage for nonlinear optimization and the difficulties that arise when working with nonlinear functions. Both constrained and unconstrained methods will be discussed.

In Chapter Four we will examine how Newton's root-finding method is derived and its relevance to the optimization of a single

variable function. A similar approach will be taken with Newton's multivariable system of equations method and then applied to the goal of optimization of a nonlinear function of several variables. In particular we will be discussing the minimization of a multivariable nonlinear function. The theorem that most of this work is based upon will be presented and then translated to fit the goals of minimization.

Next we will examine four functions to which the theorem has been applied and compare the results of both of the Newton algorithms covered as well as compare the paths taken with those of steepest descent methods.

Mathematica, a general software system for mathematical computations, has been used to perform both the necessary calculations and for the generation of the graphs presented in this thesis [15]. A user defined algorithm named *findRoot* that applies Newton's method to a system of multivariable equations has also been used and its attributes compared to the function FindRoot from *Mathematica.* A user defined algorithm named *gradsearch,* which applies a steepest descent method, has been used and its attributes and paths compared with those of the FindMinimum function from *Mathematica.*

CHAPTER II

NOTATION, DEFINITIONS AND PRELIMINARIES

NOTATION

The following notational policies will be followed throughout this thesis:

- Superscripts will be used to represent exponents. For example,

  x^2 represents x being raised to the 2^{nd} power.

- Superscripts in parentheses will be used to represent the iteration number. For example,

  $x^{(k)}$ represents the k^{th} iterate.

- Subscripts will be used to represent the components of a vector. For example,

  x_k represents the k^{th} component of the $\mathbf{x}$ vector.

Throughout this thesis a boldfaced lower-case letter $\mathbf{x}$ will represent a vector, and all vectors will be stated as column vectors. Boldfaced capital letters, such as $\mathbf{J}$, will represent matrices.

OPTIMIZATION TERMS

Decision variable: a variable that represents the quantifiable decisions to be made and whose value will be determined by the programming problem.

Objective function: a function, written in terms of the decision variables, that mathematically represents the goals and objectives of the programming problem.

Feasible solution: a solution for which all of the constraints are satisfied.

Functional constraints: mathematical equations and inequalities that represent restrictions/constraints on the decision variables and determine the feasible region in which a feasible solution must lie.

Infeasible solution: a solution for which at least one constraint is violated.

Optimal solution: a feasible solution that has the most favorable value of the objective function whether it is in maximizing or minimizing form. It will be denoted by $\mathbf{x}^*$.

Throughout the thesis the term global will be used to denote a method that is designed to converge to a local minimizer or maximizer of a nonlinear function or to a solution of a system of nonlinear equations from almost any starting point. The term local or local convergence will be used to denote a method that is designed to converge to a local minimizer or maximizer from a point sufficiently close to it.

MATHEMATICAL BACKGROUND

For $F: R^n \to R^n$ the Jacobian matrix of F at $\mathbf{x}$ is the matrix whose i, j element is

$$\mathbf{J}(\mathbf{x})_{ij} = \frac{\partial F_i(\mathbf{x})}{\partial x_j} \qquad 1 \le i, j \le n.$$

The following definitions hold for a twice continuously differentiable function $f: R^n \to R$:

The gradient vector of the function is defined as

$$\nabla f(x_1, x_2, \ldots, x_n) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]^T.$$

The Hessian matrix is an $n \times n$ matrix whose i, j element is

$$\nabla^2 f(\mathbf{x})_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \quad 1 \le i, j \le n \text{ and will be denoted as } \mathbf{H}_f.$$

A symmetric $n \times n$ matrix $\mathbf{A}$ is said to be positive definite if for all nonzero $n \times 1$ vectors $\mathbf{x}$, $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ and is positive semidefinite if $\mathbf{x}^T\mathbf{A}\mathbf{x} \ge \mathbf{0}$ for all vectors $\mathbf{x}$. Similarly a symmetric matrix is negative definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} < 0$ for all nonzero vectors $\mathbf{x}$ and negative semidefinite if $\mathbf{x}^T\mathbf{A}\mathbf{x} \le 0$ for all vectors $\mathbf{x}$.

Let $\mathbf{A}$ be an $n \times n$ matrix with real or complex components. The number λ is called an eigenvalue of $\mathbf{A}$ if there is a nonzero vector $\mathbf{v}$ in C^n such that $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$. The vector $\mathbf{v} \neq \mathbf{0}$ is called an eigenvector of $\mathbf{A}$ corresponding to the eigenvalue λ.

Each of the following gives necessary and sufficient conditions for the symmetric matrix $\mathbf{A}$ to be positive definite:

1. all of the eigenvalues λ of $\mathbf{A}$ satisfy $\lambda > 0$
2. all of the leading principal submatrices $\mathbf{A}_k$ (the k x k matrix in the upper left-hand corner of $\mathbf{A}$) have positive determinants.

The spectral radius, $\rho(\mathbf{A})$ of a matrix $\mathbf{A}$ is defined as $\rho(\mathbf{A}) = \max|\lambda|$ where λ is an eigenvalue of $\mathbf{A}$.

The function $f : \mathrm{R}^n \to \mathrm{R}$ is a convex function if, for each pair of points on the graph of f, the line segment joining these two points lies entirely above or on the graph of f. It is strictly convex if this line segment actually lies entirely above the graph except at the endpoints of the line segment. A twice continuously differentiable function f is convex when its $n \times n$ Hessian matrix

is positive semidefinite for all possible values of $(x_1, x_2, \ldots, x_n)^T$. The function is strictly convex if the Hessian is positive definite at all values of $(x_1, x_2, \ldots, x_n)^T$.

Similarly a function f is concave if, for each pair of points on the graph of f, the line segment joining these two points lies entirely below or on the graph of f. It is a strictly concave function if this line segment actually lies entirely below the graph except at the endpoints of the line segment. A twice continuously differentiable function f is concave if its $n \times n$ Hessian matrix is negative semidefinite for all possible values of $(x_1, x_2, \ldots, x_n)^T$ and is strictly concave if the Hessian is negative definite at all values of $(x_1, x_2, \ldots, x_n)^T$.

The vector norms that have been used for the analysis and research involved with this thesis are as follows:

1. $\|\mathbf{v}\|_2 = (\sum_{i=1}^{n} v_i^2)^{1/2}$ which is the Euclidean or l_2 norm.
2. $\|\mathbf{v}\|_\infty = \max_{1 \le i \le n} |v_i|$ which is called the l_∞ norm or sup norm.
3. $\|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i|$ which is called the l_1 norm.

The matrix norms that are induced by the l_2 and l_∞ norms are:

1. $\|\mathbf{A}\|_2 = (\rho(\mathbf{A}^T\mathbf{A}))^{1/2}$ induced by the l_2 norm

   (If for a matrix $\mathbf{A}$, $\mathbf{A}^T = \mathbf{A}$ then $\|\mathbf{A}\|_2 = \rho(\mathbf{A})$.)
2. $\|\mathbf{A}\|_\infty = \max_{1 \le i \le n} \{ \|a_{i.}^T\|_1 \}$ induced by the l_∞ norm, where a_i is the i^{th} row of $\mathbf{A}$.

A function $g : X \to Y$ is Lipschitz continuous with constant γ in the set X, written $g \in Lip_\gamma(X)$, if for every $x, y \in X$,

$\|g(x) - g(y)\|_Y \le \lambda \|x - y\|_X$ where $\|.\|_X$ and $\|.\|_Y$ are norms on the spaces X and Y.

If there exists a $p \geq 1$ and c such that

$$\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\| \leq c \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|^p \qquad \text{(where } c < 1 \text{ if } p = 1\text{)}$$

and $\mathbf{x}^{(k)} \to \mathbf{x}^*$ as $k \to \infty$, then the order of convergence of the sequence $\{\mathbf{x}^{(k)}\}$ $k = 1,2,\ldots$ is at least p. If $p = 1$, the sequence converges linearly. If $p = 2$, then the sequence converges quadratically.

The notation $\mathrm{N}(\mathbf{v}, r)$ will be used to describe the open neighborhood of radius r centered at the vector $\mathbf{v}$, i.e. $\mathrm{N}(\mathbf{v}, r) = \{\mathbf{w} \mid \|\mathbf{w} - \mathbf{v}\| < r\}$.

Throughout this thesis, we have chosen to work with the l_∞ and the l_2 norms. The following relationship exists between these vector norms:

$$\|\mathbf{v}\|_\infty \leq \|\mathbf{v}\|_2 \leq \sqrt{n}\|\mathbf{v}\|_\infty \tag{2.1}$$

with n representing the dimension of the space R^n.

For $n \times n$ matrices a similar relationship exists:

$$\frac{1}{\sqrt{n}}\|\mathbf{A}\|_\infty \leq \|\mathbf{A}\|_2 \leq \sqrt{n}\|\mathbf{A}\|_\infty . \tag{2.2}$$

We make use of the triangle inequality which states that

$$\|\phi + \psi\| \leq \|\phi\| + \|\psi\| \text{ for } \phi, \psi \text{ in a normed space.} \tag{2.3}$$

Another useful inequality is

$$|a| + |b| \leq 2Max\{|a|, |b|\} \text{ where } a, b \in \mathrm{R}. \tag{2.4}$$

CHAPTER III

HISTORY AND BACKGROUND

Whether under the guise of optimization, mathematical programming, management science or a host of other names, the goal of operations research is to arrive at the "best possible" conclusion under a given set of circumstances.

Although the study of optimization methods can be traced as far back as the days of Newton, Lagrange and Cauchy, its true renaissance came during the troubled times of World War II when the British military forces began studying adequate ways to use limited military resources. After successful results were witnessed, the United States military began similar research activities and soon became the leader in this new discipline [13]. Following the war, the successes of military operations research were noticed by industrial managers and other leaders who dealt with issues related to limited resources, and the concepts of operations research expanded rapidly into the business/management arena [12]. With the rapid development of modern computers with increased computational abilities and storage capabilities, the study of operations research moved quickly from just military and business applications into hospitals, libraries, financial institutions and a host of other applicable areas.

As mentioned previously, the goal of operations research is to reach the "best" decision under given circumstances. The difficulty in making this determination arises in the definition of best; the best may not be attainable in all situations and what is best for one person may be worst for another. In

short, we are often unsure what best means and must seek further clarification before attempting to solve a problem.

Several strategies can be employed to determine what is best in a given situation and to offer a varying degree of realism. One such strategy is operational exercises, which involves real-life experiments from which generalizations are drawn. Although costs can be prohibitively high in order to collect sufficient data, this method offers the highest degree of realism; human interaction is maintained with very little abstraction or simplification. Another method often employed is that of gaming, whereby a model is constructed that is a simplified version of reality. Decisions are made by humans during the implementation and using these options is not as expensive as actually creating/implementing a real-life situation as in operational exercises. Another successful option is simulation. This strategy is similar to gaming in that a model is made that simulates reality, but decision inputs are made externally before evaluation. Simulation techniques are often in the form of a computer program; therefore human interaction during implementation does not occur [4].

The three methods mentioned so far do not generate alternatives and do not provide an optimal solution. They simply evaluate the input decisions with varying degrees of human interaction involved.

The fourth way of overcoming this difficulty, defining what is best, is to form a concrete idea of best –i.e., to represent "best" by a mathematical concept (function). This function should represent the goals/objectives of the problem, and any conditions under which that goal is to be achieved should be represented mathematically as well. This approach is called an analytical model. It is entirely represented in mathematical terms and offers the most abstract model of the types discussed. Use of an analytical model allows for the

generation of alternatives, although not explicitly called for, and finding an optimal solution, $\mathbf{x}^*$, that satisfies the given conditions [4]. It is also the type of modeling that will be the primary focus of this thesis.

The general optimization problem, using an analytical model, is in the following form:

$$\text{maximize or minimize } f(\mathbf{x})$$

$$\text{subject to } \quad \mathbf{x} \in \Omega \quad \text{where } \Omega \subseteq \mathrm{R}^n.$$

Given the function is continuous and the feasible region is nonempty and compact (closed and bounded) then an optimal solution will exist. With this concept in mind, the study of operations research forks into several branches; we will look at two in particular, linear and nonlinear programming. As its name implies, a linear programming problem in its standard form is made up of a linear objective function together with linear constraints. Linear programming is one of the most developed and widely used branches in operations research. The standard form for a linear programming problem follows:

$$\text{maximize or minimize } f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \quad g_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{for } i = 1,2,\ldots,m$$

$$\text{and } x_j \ge 0 \quad \text{for } j = 1,2,\ldots,n.$$

With both a linear objective function and linear constraints, this type of problem offers several simplifications on the existence of an optimal solution and its location if one does exist. As in the general case, requiring that the objective function be a continuous (linear) function and that the feasible region be closed and bounded guarantees an optimal solution. The simplicity of linear programming comes from the geometric interpretation of maximizing or

minimizing a linear function. We are guaranteed that an optimal solution exists and lies at an extreme point of that feasible region --i.e., at a corner point (the intersection of two or more constraints). If more than one optimal corner-point solution exists, then every convex combination of the optimal corner point solutions will be an optimal solution as well, i.e., $\sum_{k=1}^{m} \lambda_k \alpha_k$ where α_k represents the k^{th} optimal corner point and $\lambda_k \geq 0$ with $\lambda_1 + \lambda_2 + \ldots + \lambda_m = 1$. Therefore, we need check only the corner points of the feasible region and not the entire feasible region. It is this point that led to the development of the simplex method (1947 Dantzig)—a highly successful method for solving the general linear programming problem [5]. It is a multipurpose algorithm that can handle the typical linear programming (LP) problem and is not too difficult to use. The basic idea behind the simplex method is move from one corner-point feasible solution to the next and to evaluate the objective function there. It continues to move until no further improvement can be made; in essence, it does a boundary trace although some regions of the boundary may never be reached.

One of the benefits of the simplex method is that is can solve any linear programming problem and in a finite number of iterations, given that precautions are used to prevent cycling [6, page 33]. It deletes redundant constraints, identifies when the objective function is unbounded, and identifies when there is more than one optimal solution. Not only does this method provide an optimal solution but also it offers much more information along with that solution. Sensitivity analysis can be easily performed to determine how the optimal solution would vary with changes in the problem data.

Other methods have been studied for linear programming problems, especially huge linear programming problems. One such method is the interior

point algorithm developed by Narendra Karmarkar [10]. The idea behind this method is to shoot through the interior of the feasible region toward the optimal solution as opposed to a boundary trace which is done in the simplex method. The method moves in a direction that improves the objective function at the fastest possible rate. Interior point algorithms transform the feasible region, at each iteration, to place the current trial solution near its center. This method is used primarily on very large linear problems for which even a boundary trace may prove to be extremely time consuming.

The second branch of operations research we will discuss is nonlinear programming, occurring when one or more of the functions needed to define the program is nonlinear in nature. In one general form, the nonlinear programming problem is to find $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ in order to

$$\text{maximize or minimize } f(\mathbf{x})$$

$$\text{subject to } g_i(\mathbf{x}) \le b_i \text{ for } i = 1,2,\ldots,m$$

$$\text{and } \mathbf{x} \ge \mathbf{0}.$$

With nonlinear programming, we no longer have the simplifications that occurred in linear programming, mainly that we need only consider a limited number of feasible solutions. Without this guarantee, we must consider all of the feasible points both inside the feasible region and on its boundary. It is not always possible to reach an optimal solution; instead we may simply have to approach the optimal solution.

With the various kinds of nonlinearities that can occur, there is no longer an all-purpose algorithm that can handle all of these differences. With nonlinear functions, even if one is able to find a local extremum, there is no guarantee that the same point will be a global maximizer or minimizer except under given conditions. Therefore it becomes necessary to be able to determine under what

conditions a local extremum will be guaranteed to be a global maximizer or minimizer, hence an optimal solution.

Nonlinear optimization has two main branches, constrained and unconstrained optimization. We will briefly look at constrained nonlinear optimization, both convex and non-convex, and at some of the most commonly used algorithms. Then we will consider unconstrained optimization in detail focusing primarily on the Newton, quasi-Newton methods and gradient methods.

Under given circumstances, a local extremum can be guaranteed to be a global maximizer or minimizer depending upon whether we are maximizing or minimizing the given function. The requirements for a maximization problem are that there is a concave objective function and a convex feasible region (meaning, in general, that all of the functions used in the constraints are convex). For minimization, the only alteration needed would be to require that f be convex. Given these attributes, an optimal solution can be guaranteed. For ease in exposition, we will consider the minimization case from this point forward.

The Karush-Kuhn-Tucker (KKT) conditions offer a way of determining if a solution is optimal for the general constrained optimization problem [10]. They are a set of equations and inequalities that must be satisfied for any multivariable constrained optimization problem in order for a solution to be considered optimal. In other words, these conditions are necessary but not sufficient conditions for optimality. If the nonlinear programming problem is a convex programming problem, then satisfying the KKT conditions becomes a sufficient condition for optimality.

Although it is often difficult to solve for $\mathbf{x}^*$ directly from the KKT conditions, it can sometimes be done. More often, however, the KKT

conditions are applied indirectly to define a dual problem or to rewrite quadratic programming problems (problems with linear constraints and a quadratic objective function) in a linear programming form. The idea is to use the KKT conditions as linear constraints and then restrict the choices allowed in the simplex method in a way that will include the constraints created by the KKT conditions. The linear program created can then be solved by modifying the simplex method and is adequately named the modified simplex method [14].

An often used tactic for solving a constrained nonlinear optimization problem is to try and reformulate the model as a linear program. Approximating nonlinear objective functions and/or functional constraints by linear functions will often accomplish this. One such method is the separable programming method [10]. The idea behind this method is to take a nonlinear, separable convex program (a program where each term of the objective function and the constraint functions involves a single variable) and approximate it by a linear programming problem with a larger number of variables. It does so by creating a piecewise linear function which gets rewritten as a linear function of several variables (each new variable represents a line segment) and adds to this set-up one restriction on the use of these new variables. The added restriction is that the next variable in line (i.e., the next segment) cannot be used until the previous variable has reached its upper limit. In terms of resources the meaning would be that one could not use a new resource until all of the previous resources have been exhausted. The benefit of this type of reformulation is that the simplex method can then be used to solve the linear program, and the optimal solution to the original problem can be approximated from the optimal solution to the linear programming problem. One of the drawbacks to approximating functions by piecewise linear

functions is that to get a "good" approximation may require a large number of variables.

Another method that uses a linear approximation for the objective function is the Frank-Wolfe method [10]. This method can be applied to linearly constrained convex programming problems; it uses a linear approximation of the objective function combined with a one-dimensional search method to obtain a sequence of trial solutions that converges to the optimal solution. For the linear approximation of the objective function, this method uses the first-order Taylor expansions and minimizes this new objective function subject to the original constraints. The trial solutions for this method approach the optimal solution alternately between two or more trajectories and tend to converge rather slowly. To help speed up the convergence, we can extrapolate an estimate of the intersections of these trajectories and find a closer approximation than the current trial solution.

Yet another approach for solving constrained nonlinear programming problems, and the one that is most related to the focus of this thesis, is to solve sequences of unconstrained problems whose solutions converge to the optimal solution of the original constrained problem. Sequential unconstrained minimization techniques, SUMT, often called penalty or barrier techniques, can be split into two categories: interior and exterior methods. Interior methods allow the trial solutions to approach the optimal solution from the interior of the feasible region; exterior methods approach the optimal solution from outside the feasible region [13]. The general idea behind both interior and exterior methods is to incorporate the functional constraints into the objective function in a way that will add or subtract a huge penalty to the objective function when the trial solution is near the boundaries of the feasible region. The iterative process allows for the scaling factor to control the penalty when

the optimal solution may lie near or on a boundary. As the penalty (barrier) function goes to zero, the new objective function converges to the original objective function. This method can be used for both convex and non-convex programming problems as well as for both minimization and maximization and can be adapted to handle both inequality and equality constraints.

For many constrained optimization problems the optimal solution can be found by solving a sequence of unconstrained optimization problems. We will be discussing line search methods such as the gradient search (steepest descent) method and Newton/quasi-Newton methods with the focus being primarily on Newton's method.

The general description of a descent method is as follows:

1. Start at some given point $\mathbf{x}^{(0)}$
2. Assign $k = 0$
3. Choose a search direction $\mathbf{d}^{(k)}$.
4. Use a line minimization method (one-dimensional search method) to minimize $f(\mathbf{x}^{(k)} + s^{(k)}\mathbf{d}^{(k)})$ by varying the step-length $s^{(k)}$
5. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + s^{(k)}\mathbf{d}^{(k)}$
6. Have we converged? (Various criteria can be applied to perform such a test.)
   a) if yes, output result
   b) if no, increment k and go back to step 3.

As seen above, the iterative form of the general descent problem is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + s(\mathbf{x}^{(k)})\mathbf{d}(\mathbf{x}^{(k)})$$

where $s(\mathbf{x}^{(k)})$ is the step length determined at the k^{th} iterate and $\mathbf{d}(\mathbf{x}^{(k)})$ is the search direction determined at the k^{th} iterate. In the steepest descent method the choice of which direction to move is the direction of the negative gradient, i.e. $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$ and for Newton's method the direction of choice is

$-\mathbf{H}_f(\mathbf{x}^{(k)})\nabla f(\mathbf{x}^{(k)})$ and $s^{(k)} = 1$ [11]. When the full Newton step is not adequate some adaptation of the step length is necessary. Such methods are called quasi-Newton methods and include ideas such as "backtracking line searches", which take a modified step in the Newton direction, and "model trust-regions" and "hook" methods, which choose a step length first and then determine the direction in which to move. Checking that $\mathbf{H}_f(\mathbf{x}^{(k)})$ is positive definite at $\mathbf{x}^{(k)}$ ensures the direction $\mathbf{d}^{(k)}$ chosen will be a descent direction.

One way to look at the steepest descent method, in the two-dimensional case, is to think of the gradient indicating the direction that water would flow downhill. This method uses the gradient to determine the direction of greatest improvement, in this case minimizing the objective function, and then, a one-dimensional search method to determine how far to move in that direction. It then continues in that direction until no further improvement can be made. The direction of the steepest descent is a local property and not a global property. Therefore the gradient must be recalculated; the procedure is then followed iteratively until a point is reached where there is essentially no decrease in the objective function (i.e., until each partial derivative of f with respect to each component of $\mathbf{x}$ is less than a selected error tolerance). This stopping criterion is using the l_∞ norm and is just one of many stopping criteria that may be used. The steepest descent method is considered a first-order method of optimization because it relies only on first-derivative information and has a linear convergence rate.

In the two-dimensional case the method of steepest descent settles into a steady zigzag of parallel and perpendicular segments and converges only linearly although some modifications can be made to speed up its convergence [13]. For higher dimensions the segments may not alternately be parallel and perpendicular but each pair of successive segments is perpendicular. In other

words, the first and third segments may not be parallel to each other but both are perpendicular to the second segment. It is easy to establish that the direction in which the greatest increase in f occurs is in the direction of the gradient. For minimization, we would need the direction of greatest decrease that occurs in the direction of the negative gradient. Using the steepest descent method for minimization, a step length is found that determines how far along the negative gradient one can move and still find improvement stopping at some point $\mathbf{p}$. Any further movement in that direction will bring no improvement of the function value. Therefore, the choice for the next move is again in the direction of the negative gradient in order to achieve the greatest improvement of f. Each successive move is in a direction perpendicular to the previous move.

When the problem is sufficiently small (in terms of the number of variables) this method can be used alone to solve a minimization problem; more often, however, it is used to find a "good" approximation to begin a more rapid locally convergent method such as Newton or quasi-Newton methods.

CHAPTER IV

NEWTON METHODS

We will begin our discussion of Newton's method by considering the problem of finding the roots of a nonlinear equation $f(x)=0$ where $x \in \mathrm{R}$. Although the importance may not be clear immediately, the connection to optimization will be seen at a later point in the thesis. One of the most powerful and well-known numerical methods for solving a root-finding problem is Newton's method. We will derive Newton's method based on using the Taylor polynomial expansions of f.

Taking the Taylor expansion of the function f at a point $\bar{x}$ not far from a solution x^* (i.e. $|\bar{x}-x^*|$ is "small") gives

$$f(x)=f(\bar{x})+(x-\bar{x})f'(\bar{x})+\frac{(x-\bar{x})^2}{2}f''(\xi(x))$$

where $\xi(x)$ lies between x and $\bar{x}$. Since $f(x^*)=0$ at the solution, when $x=x^*$ the expansion becomes

$$0=f(\bar{x})+(x^*-\bar{x})f'(\bar{x})+\frac{(x^*-\bar{x})^2}{2}f''(\xi(x^*)).$$

Newton's method is derived by assuming that since $|x^*-\bar{x}|$ is small, the term involving its square is negligible so that the above equation is approximated by

$$0 \approx f(\bar{x})+(x^*-\bar{x})f'(\bar{x})$$

and therefore, solving for x^* gives

$$x^* \approx \bar{x} - \frac{f(\bar{x})}{f'(\bar{x})}$$

and for the iterative process is described as

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

For most problems, Newton's method will converge quadratically to a root of one nonlinear equation in one unknown provided it is given a "good enough" starting point [7].

Theorem 2.4.3 in [7] states the conditions under which Newton's method is guaranteed to converge and gives the rate at which it converges given a "good" estimate:

THEOREM 4.1

Let $f : \mathrm{D} \to \mathrm{R}$, for an open interval D, and let $f' \in Lip_\gamma(\mathrm{D})$. Assume that for some $\rho > 0$, $|f'(x)| \geq \rho$ for every $x \in \mathrm{D}$. If $f(x) = 0$ has a solution $x^* \in \mathrm{D}$, then there is some $\eta > 0$ such that: if $|x^{(0)} - x^*| < \eta$, then the sequence $\{x^{(k)}\}$ generated by

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \qquad k = 0,1,2,...$$

exists and converges to x^*. Furthermore, for $k = 0,1,...,$

$$|x^{(k+1)} - x^*| \leq \frac{\gamma}{2\rho}|x^{(k)} - x^*|^2.$$

Stating that $|f'(x)| \geq \rho$ gives the first derivative a lower bound and states that the first derivative must be nonzero at x^* for Newton's method to converge quadratically. Otherwise the method is not guaranteed to converge quadratically and would converge linearly (see [2], for example).

The Lipschitz constant γ is considered a measure of the nonlinearity of the function f. However it is dependent upon scaling. Another measure of nonlinearity can be obtained by dividing γ by $f'(x)$. The theorem states that the smaller this measure of relative nonlinearity is, the faster Newton's method will converge to the optimal solution. One thing that must be pointed out is that Newton's method can be guaranteed to converge from a "good" starting point but nothing can be guaranteed from a "bad" starting point. Therefore Newton's method is a locally convergent method but not a globally convergent method and must be adapted and used in conjunction with a globally convergent method to be successful from other starting points.

Newton's method for a single variable equation determines where a zero of a function can be found. To apply Newton's method as an optimizer for a single variable function the same method is applied to the derivative of the function, solving for $f'(x) = 0$ in which case the iterative equation becomes

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

This equation is formed by taking a linear approximation of f' and is equivalent to making a quadratic approximation of the original function f.

This quadratic approximation is much more suited to minimization because a quadratic function has only one extremum, meaning that Newton's method will solve for the optimizer of a quadratic function is one step. However in most cases Newton's method must be applied iteratively to approximate the root of the function. The minimization of a single variable function possesses the same convergence properties as a single variable equation, and again its guarantees come from starting at a point that is sufficiently close to x^*. Therefore a large amount of effort is often made in finding a point "good enough" to guarantee success with Newton's method. One way to find such a

point is to use a globally convergent method to find a point in a "good" neighborhood for which Newton's method can be successfully started.

It should be stated that all of the applications of Newton's method discussed so far have involved the calculation of the derivatives of a function. It is not always desirable to calculate the derivatives nor is it always possible. At these times, a secant method can be used to approximate these derivatives using only function evaluations, and doing so requires two points at each iteration instead of just one. One might choose to use these approximations if the cost of finding the derivatives is high or when the evaluation of the derivatives is too time-consuming.

We have considered the application of Newton's method to a single-variable equation and the application of Newton's method to the minimization of a single-variable function. Now we will proceed to the use of Newton's method to solve a multivariable system of equations and examine how it can be applied to the optimization of a multivariable nonlinear function.

The general form of a system of nonlinear equations is as follows:

Given $\quad F: R^n \to R^n$ find $\mathbf{x}^* \in R^n$ such that $F(\mathbf{x}^*) = \mathbf{0}$.

When applying Newton's method to this system of equations we again need to find a root of an affine approximation to F at the current iterate $\mathbf{x}^{(k)}$. The first partial derivatives will form the Jacobian matrix $\mathbf{J}(\mathbf{x})$ so the iterative form used will be

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)}).$$

It will be necessary to know that the Jacobian matrix is nonsingular at this point. As in the single variable case, there may be times when the Jacobian is not analytically available, and it will be necessary to use an approximation of this matrix.

The application of Newton's method to the optimization of a multivariable nonlinear function is of key importance to the work in this thesis. In general form the optimization of a multivariable nonlinear function is as follows:

minimize $f(\mathbf{x})$ where $f:\mathrm{R}^n \to \mathrm{R}$.

Again Newton's method is derived from the affine approximation of ∇f using the Taylor expansion of the function around a given point and solving for $\nabla f = \mathbf{0}$. Taking the Taylor expansion of the function ∇f at a point $\bar{\mathbf{x}}$ not far from $\mathbf{x}^*$ (i.e. $\|\bar{\mathbf{x}} - \mathbf{x}^*\|$ is small) gives

$\nabla f(\mathbf{x}) = \nabla f(\bar{\mathbf{x}}) + \mathbf{H}_f(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}}) + \frac{1}{2}\phi(f, \mathbf{x} - \bar{\mathbf{x}})\,|_{\xi}$. where ξ lies on the line segment between $\bar{\mathbf{x}}$ and $\mathbf{x}$. The error term consists of the vector $\phi(f, \mathbf{x} - \bar{\mathbf{x}})$ which is computed from the vector $\mathbf{x} - \bar{\mathbf{x}}$ and a multimatrix (a higher dimensional analog of a matrix) of third order partial derivatives of f. Using the affine approximation gives

$$\nabla f(\mathbf{x}) \approx \nabla f(\bar{\mathbf{x}}) + \mathbf{H}_f(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}}).$$

Since $\nabla f(\mathbf{x}^*) = \mathbf{0}$ at the solution, when $\bar{\mathbf{x}} = \mathbf{x}^*$ the approximation becomes

$$\mathbf{0} \approx \nabla f(\bar{\mathbf{x}}) + \mathbf{H}_f(\bar{\mathbf{x}})(\mathbf{x}^* - \bar{\mathbf{x}})$$

and therefore, solving for $\mathbf{x}^*$ gives

$$\mathbf{x}^* \approx \bar{\mathbf{x}} - \mathbf{H}_f(\bar{\mathbf{x}})^{-1}\nabla f(\bar{\mathbf{x}})$$

and for the iterative process is described as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{H}_f(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)}).$$

As mentioned previously, this method requires the calculation and inversion of the second-derivative Hessian matrix and is considered a second-order method. Again the derivatives may not be analytically available or may be too time-consuming to calculate, and approximations can be made. The problem with these calculations is in storing and inverting this matrix for large

systems and in practice the Hessian may not be inverted but a linear system solved instead. Since second derivatives generally change relatively slowly, in practice the Hessian may not be recalculated at each iteration but may be done so periodically.

As seen above, this method can converge to any point where the gradient is zero, so it will just as happily settle on a minimizer, maximizer or a saddle point. For small problems such as functions of two variables like the functions discussed in this thesis, determining if the Hessian is positive definite at the critical point by checking the determinants of the leading principal submatrices and the determinant of the Hessian is sufficient. For real-life applications, with many more variables, these calculations would be costly and should be avoided. In practice, a more commonly used method is to perform a matrix factorization which can be done only if the matrix is positive definite. In the event the matrix is not positive definite, perturbations can be made to the Hessian to make it safely positive definite and the algorithm can then be applied [7, Section 5.5].

As in the case of solving a single equation in one unknown, certain conditions will guarantee the convergence of Newton's method in the nonlinear systems case. Theorem 5.2.1 in [7] states the requirements for Newton's method to converge and the rate at which it converges:

THEOREM 4.2

Let $F: R^n \to R^n$ be continuously differentiable in an open convex set $D \subseteq R^n$. Assume that there exists $\mathbf{x}^* \in R^n$ and $r, \beta > 0$, such that $N(\mathbf{x}^*, r) \subseteq D$, $F(\mathbf{x}^*) = \mathbf{0}$, $\mathbf{J}(\mathbf{x}^*)^{-1}$ exists and $\left\|\mathbf{J}(\mathbf{x}^*)^{-1}\right\| \leq \beta$, and $\mathbf{J} \in Lip_\gamma(N(\mathbf{x}^*, r))$. Then there exists $\varepsilon > 0$ such that for all $\mathbf{x}^{(0)} \in N(\mathbf{x}^*, \varepsilon)$ the sequence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ generated by

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}(\mathbf{x}^{(k)})^{-1} \mathrm{F}(\mathbf{x}^{(k)})$, $k = 0,1,...$ is well defined, converges to $\mathbf{x}^*$, and obeys $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \le \beta\gamma \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2$, $k = 0,1,...$

A value for ε must be used in the application of this theorem; the value chosen in the proof of this theorem in [7] is

$$\varepsilon = \min\left\{r, \frac{1}{2\beta\gamma}\right\} \tag{4.1}$$

where β, γ are defined in Theorem 4.2.

When reformulated for the optimization problem, Theorem 4.2 leads to the following:

THEOREM 4.3

Let $f : \mathrm{R}^n \to \mathrm{R}$ be twice continuously differentiable in an open convex set $\mathrm{D} \subseteq \mathrm{R}^n$. Assume that there exists $\mathbf{x}^* \in \mathrm{R}^n$ and $r, \beta > 0$ such that $\mathrm{N}(\mathbf{x}^*, r) \subseteq \mathrm{D}$, $\nabla f(\mathbf{x}^*) = \mathbf{0}$, and $\mathbf{H}_f(\mathbf{x}^*)^{-1}$ exists with $\left\|\mathbf{H}_f(\mathbf{x}^*)^{-1}\right\| \le \beta$ and $\mathbf{H}_f \in Lip_\gamma(\mathrm{N}(\mathbf{x}^*, r))$. Then there exists $\varepsilon > 0$ such that for all $\mathbf{x}^{(0)} \in \mathrm{N}(\mathbf{x}^*, \varepsilon)$ the sequence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ...$ generated by $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{H}_f(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)})$ is well defined, converges to $\mathbf{x}^*$ and obeys $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \le \beta\gamma \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2$ for $k = 0,1,2,....$

The additional condition necessary to guarantee quadratic convergence to a minimizer is to require that the Hessian, when evaluated at the optimal point $\mathbf{x}^*$, be positive definite. Otherwise the theorem guarantees quadratic convergence to a critical point only and for optimization we are interested specifically in the maxima and minima for a particular function. In particular, we will use P_m to represent the largest open convex set surrounding the m^{th} critical point on which the Hessian is positive definite.

Some of the advantages and disadvantages of Newton's method are provided in the following tables [7].

ADVANTAGES
1. Quadratically convergent from a good starting point if $\mathbf{J}(\mathbf{x}^*)$ is nonsingular for the nonlinear systems case and if $\mathbf{H}_f(\mathbf{x}^*)$ is positive definite for the minimization problem.
2. Provides the exact solution in one iteration for an affine F (exact at each iteration for any affine component of F) for the nonlinear systems case and does so if the function f is quadratic for the nonlinear optimization case.

DISADVANTAGES
1. Not globally convergent for many problems.
2. Requires the calculation of $\mathbf{J}(\mathbf{x}^{(k)})$ at each iteration for nonlinear systems and $\mathbf{H}_f(\mathbf{x}^{(k)})$ for nonlinear optimization.
3. Each iteration requires the solution of a system of linear equations that may be singular or ill conditioned. For nonlinear optimization the Hessian matrix may not be positive definite at each iterate.

As noted above, one of the biggest disadvantages of Newton methods is that they are not globally convergent but do converge rapidly in a "good" local neighborhood of the minimizer. One way to overcome this difficulty is to use a globally convergent method such as steepest descent to find a point sufficiently close from which Newton's method can be started. These methods are called quasi-Newton methods and use a basic Newton approach with adjustments such as altering the step length being made to achieve the benefits from both Newton's method and one of the globally convergent methods as well [7]. In order to use the best qualities of the steepest descent method and be able to obtain global convergence of Newton's method a quasi-Newton method must be used. In other words, a globally convergent method is used to reach a "good" starting range for Newton's method which we know will converge rapidly when given a "good" starting point. Together these give a globally convergent method that guarantees fast local convergence.

Several numerical examples will be useful in considering the criteria under which Theorem 4.3 allows for quadratic convergence, in particular to a minimizer of a function. We must however keep in mind that the theorem lists conditions upon which we can expect quadratic convergence and is, in essence, giving a worse case scenario for defining the neighborhood in which quadratic convergence is guaranteed to occur.

For convenience, we have chosen to use both the l_∞ and the l_2 norms and to graph all of the neighborhoods related to these examples with distance being defined by the l_2 norm (using *Mathematica* packages from [3]). The relationships (2.1) and (2.2) allow for a smooth transition from one norm to the other. Suppose we are able to establish the relationship

$$\left\|\mathbf{H}_f(\mathbf{x})-\mathbf{H}_f(\mathbf{y})\right\|_\infty \le C_1\left\|\mathbf{x}-\mathbf{y}\right\|_\infty . \tag{4.2}$$

Using (2.2), inequality (4.2) implies that

$$\left\|\mathbf{H}_f(\mathbf{x})-\mathbf{H}_f(\mathbf{y})\right\|_2 \le \sqrt{n}\left\|\mathbf{H}_f(\mathbf{x})-\mathbf{H}_f(\mathbf{y})\right\|_\infty \le \sqrt{n}C_1\left\|\mathbf{x}-\mathbf{y}\right\|_\infty .$$

Applying (2.1) yields

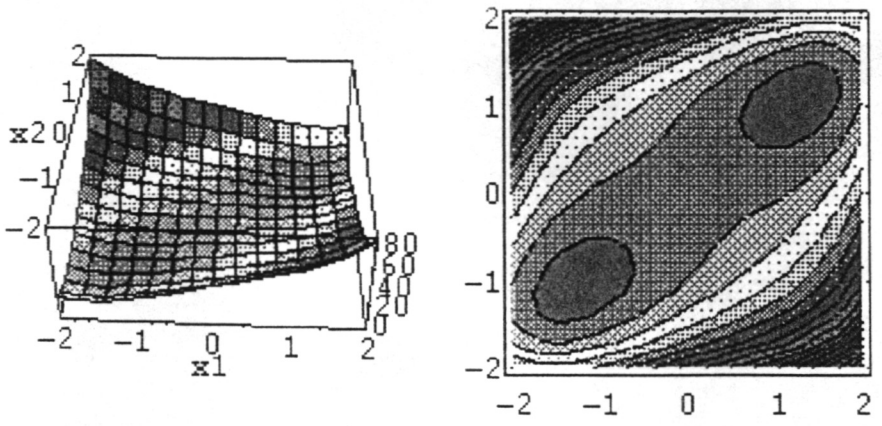
$$\left\|\mathbf{H}_f(\mathbf{x})-\mathbf{H}_f(\mathbf{y})\right\|_2 \le \sqrt{n}C_1\left\|\mathbf{x}-\mathbf{y}\right\|_\infty \le \sqrt{n}C_1\left\|\mathbf{x}-\mathbf{y}\right\|_2 . \tag{4.3}$$

For each of the examples considered in this thesis the function is twice continuously differentiable on the entire plane; however not all of the functions have Hessians which satisfy Lipschitz conditions throughout the plane. A finite value for r must be used in the analysis for some of the examples; therefore we have adopted the convention that the r-neighborhoods we will be considering will be restricted to those throughout which the Hessian is nonsingular (a condition beyond those stipulated in Theorem 4.3). The rationale behind this choice is that Newton's method fails if the Hessian is singular at any given iteration. For a minimizer, we require that the r-neighborhood be contained within the appropriate set P_m.

Example 1:

Consider the function $f(x_1, x_2) = x_1^4 + 2x_2^4 - 8x_1x_2 + 2$. Figure 4.1.1 shows a 3-dimensional representation of the function and its two-dimensional contour plot. It should be noted that the contours are darker as it approaches the minima of the given function.

Figure 4.1.1 $f(x_1, x_2) = x_1^4 + 2x_2^4 - 8x_1x_2 + 2$



By solving the system of equations formed from setting the first partial derivatives with respect to each variable equal to zero the critical points of this function are found to be $(0, 0)^T$, $(2^{3/8}, 2^{1/8})^T$ and $(-2^{3/8}, -2^{1/8})^T$. The Hessian matrix for this function is $\mathbf{H}_f = \begin{bmatrix} 12x_1^2 & -8 \\ -8 & 24x_2^2 \end{bmatrix}$ and must be evaluated at each of the critical points in order to determine if it is positive definite at that point, a condition which is sufficient for a point to be a minimizer of the given function. These facts along with a few tests from calculus will determine if we

have a maximizer, a minimizer or a saddle point at each of the critical points. Table 4.1.1 includes this information.

Table 4.1.1

Critical Point	Is the Hessian positive definite at this point?	Is the Hessian singular at this point?	Minimizer, maximizer or saddle point
1. $(0,0)^T$	No	No	Saddle point
2. $(2^{3/8}, 2^{1/8})^T$	Yes	No	Minimizer
3. $(-2^{3/8}, -2^{1/8})^T$	Yes	No	Minimizer

We now determine the regions where quadratic convergence is guaranteed by Theorem 4.3. In the present problem the Hessian is nonsingular where $x_2 \neq \pm\frac{\sqrt{2}}{3|x_1|}$. We will also need to determine the largest open convex set $P_m \subseteq R^2$ containing the m^{th} critical point on which $\mathbf{H}_f(\mathbf{x})$ is positive definite. We will use the information from the analysis of Hessian matrix to accomplish this. The Hessian is positive definite under the following conditions: 1) the (1,1) entry of the Hessian is positive and 2) the determinant of $\mathbf{H}_f$ is positive. For this particular Hessian those conditions are 1) $12x_1^2 > 0$ and 2) $12x_1^2(24x_2^2) - (-8)(-8) > 0$. Since $12x_1^2$ is always positive except when $x_1 = 0$ and $288x_1^2x_2^2 - 64$ is positive when $x_2 > \frac{\sqrt{2}}{3|x_1|}$ or $x_2 < -\frac{\sqrt{2}}{3|x_1|}$, the region where the Hessian is positive definite is the intersection of the regions resulting from these two conditions (See Figure 4.1.2). To find the largest open neighborhood containing each critical point and contained in the region $\mathbf{P}_m$ we need the minimum distance from the boundary of the region where the Hessian is nonsingular and each critical point. Using critical point 2, which is $(2^{3/8}, 2^{1/8})^T$, we must minimize the function

$$g(x_1) = (x_1 - 2^{3/8})^2 + (\frac{\sqrt{2}}{3|x_1|} - 2^{1/8})^2.$$

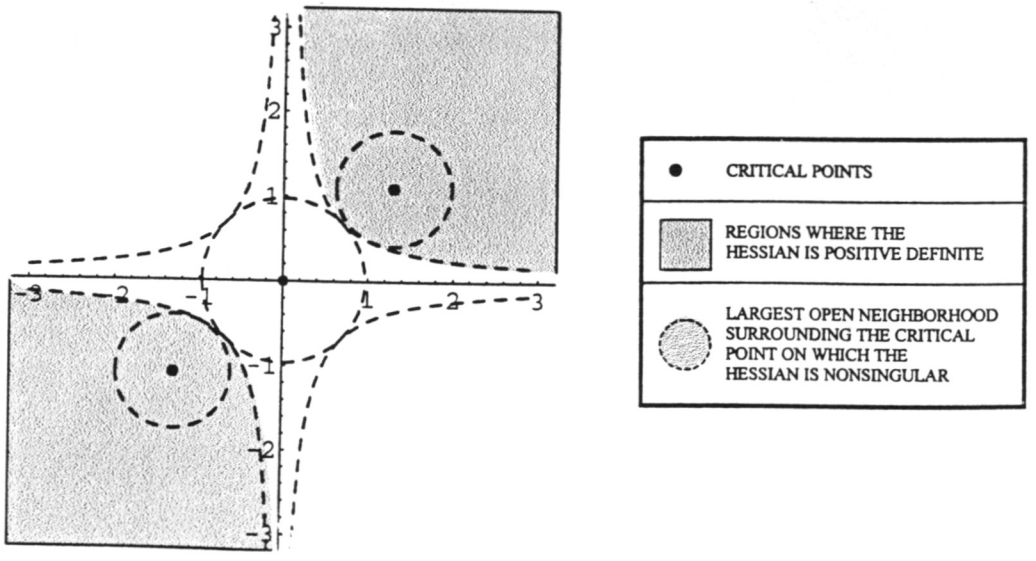
Taking the derivative of g and setting it equal to zero we obtain the critical points $x_1 = -.65954$ and $x_1 = 1.0078$. The minimum is $x_1 = 1.0078$ and gives the point $(1.0078, 0.47507)^T$, which lies on the boundary of the region. The distance between $(2^{3/8}, 2^{1/8})^T$ and $(1.0078, 0.47507)^T$ is $r = 0.67994$. Thus the largest open neighborhood containing $\mathbf{x}^*$ and contained in P_2 is $N(\mathbf{x}^*, 0.67994)$. Because of the symmetries of this function the largest open neighborhood containing critical point 3, which is $(-2^{3/8}, -2^{1/8})^T$, contained in P_3 and satisfying the conditions of Theorem 4.3 has the same radius as the previous neighborhood. Similar calculations show that the largest open neighborhood of the point $(0,0)^T$ satisfying the conditions of the theorem and the nonsingularity of the Hessian is $N((0,0)^T, 0.97098)$. (Recall that the Hessian is not positive definite at this critical point and therefore P_1 does not exist.) In the following table we have listed each critical point, the largest open neighborhood surrounding each of the critical points on which $\mathbf{H}_f(\mathbf{x})$ is nonsingular, and the largest open convex region surrounding the critical point where the Hessian is positive definite.

Table 4.1.2

Critical Point	Largest open neighborhood surrounding the critical point in which $\mathbf{H}_f$ is nonsingular	P_n = largest open convex set containing the critical point where the Hessian is positive definite
1. $(0,0)^T$	$N((0,0)^T, 0.97098)$	P_1 does not exist – the Hessian is not positive definite at this point
2. $(2^{3/8},2^{1/8})^T$	$N((2^{3/8},2^{1/8})^T, 0.67994)$	$P_2 = \left\{ \mathbf{x} \middle\| \begin{matrix} \det \mathbf{H}_f(\mathbf{x}) > 0, \\ x_1 > 0,\ x_2 > 0 \end{matrix} \right\}$
3. $(-2^{3/8},-2^{1/8})^T$	$N((-2^{3/8},-2^{1/8})^T, 0.67994)$	$P_3 = \left\{ \mathbf{x} \middle\| \begin{matrix} \det \mathbf{H}_f(\mathbf{x}) > 0, \\ x_1 < 0,\ x_2 < 0 \end{matrix} \right\}$

The information from Table 4.1.2 is represented in Figure 4.1.2.

Figure 4.1.2 Regions where the Hessian is positive definite



For the subsequent analysis of this example we will be using the minimizer $\mathbf{x}^* = (2^{3/8}, 2^{1/8})^T$. The Hessian matrix when evaluated at this point is

$$\mathbf{H}_f = \begin{bmatrix} 20.1815 & -8 \\ -8 & 28.5410 \end{bmatrix} \text{ and its inverse is } \mathbf{H}_f^{-1} = \begin{bmatrix} 0.056 & 0.016 \\ 0.016 & 0.039 \end{bmatrix}.$$

To calculate a value of β, from Theorem 4.3, we have the following inequality:

$$\| \mathbf{H}_f(\mathbf{x}^*)^{-1} \|_2 \leq \beta .$$

We have calculated the eigenvalues of $\mathbf{H}_f(\mathbf{x}^*)^{-1}$ and taken the maximum absolute value of these which equals the spectral radius of this symmetric matrix and therefore $0.072 \leq \beta$.

In order to determine the region of guaranteed quadratic convergence and the relative nonlinearity of the function f at $\mathbf{x}^*$ we will need a Lipschitz constant γ such that $\mathbf{H}_f \in Lip_\gamma(\mathrm{N}(\mathbf{x}^*, r))$, i.e. $\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \|_2 \leq \gamma \|\mathbf{x} - \mathbf{y}\|_2$ for all $\mathbf{x}, \mathbf{y} \in \mathrm{N}(\mathbf{x}^*, r)$ with $\mathbf{x}^* = (2^{3/8}, 2^{1/8})^T$.

Using the values of $\mathbf{H}_f(\mathbf{x})$ and $\mathbf{H}_f(\mathbf{y})$ for this function we obtain

$$\begin{aligned}
\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \|_\infty &= \left\| \begin{bmatrix} 12x_1^2 & -8 \\ -8 & 24x_2^2 \end{bmatrix} - \begin{bmatrix} 12y_1^2 & -8 \\ -8 & 24y_2^2 \end{bmatrix} \right\|_\infty \\
&= \left\| \begin{bmatrix} 12(x_1^2 - y_1^2) & 0 \\ 0 & 24(x_2^2 - y_2^2) \end{bmatrix} \right\|_\infty \\
&= Max\left\{ \left|12(x_1^2 - y_1^2)\right|, \left|24(x_2^2 - y_2^2)\right| \right\} \\
&\leq 24 Max\left\{ \left|x_1^2 - y_1^2\right|, \left|x_2^2 - y_2^2\right| \right\} \\
&= 24 Max\left\{ \left|x_1 + y_1\right|\left|x_1 - y_1\right|, \left|x_2 + y_2\right|\left|x_2 - y_2\right| \right\}.
\end{aligned}$$

The largest possible first coordinate in this region would be $2^{3/8} + r = 1.98340$. Similarly the largest second coordinate in this region would be $2^{1/8} + r = 1.77707$.

Using these two facts a numerical value can be substituted for each of the sums in the expression, i.e.,

$$|x_1 + y_1| \le |x_1| + |y_1| \le 2(1.98340) = 3.96680 \text{ and}$$

$$|x_2 + y_2| \le |x_2| + |y_2| \le 2(1.77707) = 3.55413.$$

Substituting these values into the expression gives

$$\begin{aligned} \left\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \right\|_\infty &\le 24 Max\left\{ 3.96680|x_1 - y_1|, 3.55414|x_2 - y_2| \right\} \\ &\le 24(3.96680) Max\left\{ |x_1 - y_1|, |x_2 - y_2| \right\} \\ &= 95.20320 \left\| \mathbf{x} - \mathbf{y} \right\|_\infty . \end{aligned}$$

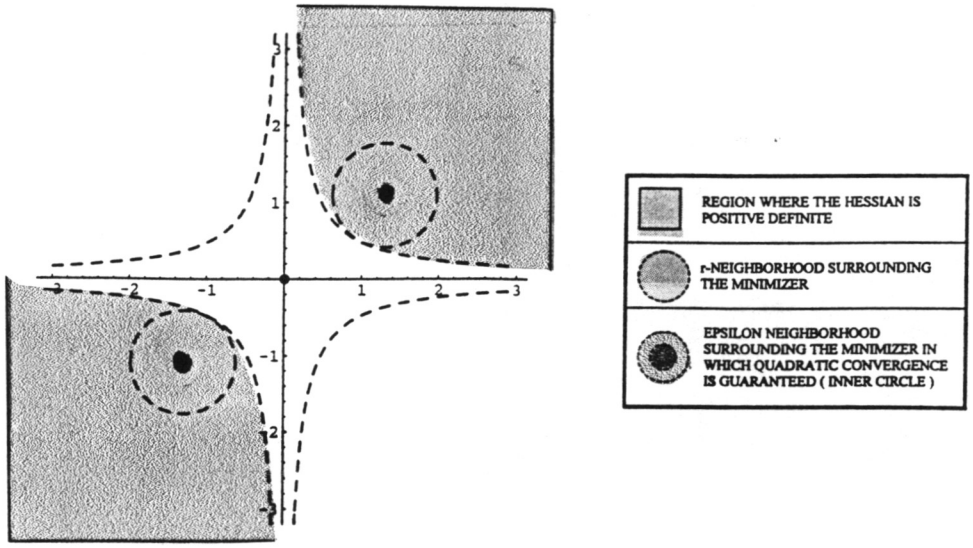
For this function $\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})$ is a diagonal matrix and therefore

$$\left\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \right\|_2 = \left\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \right\|_\infty$$

giving $\gamma = 95.20320$.

In order to find the largest open neighborhood in which for all $\mathbf{x}^{(0)} \in N(\mathbf{x}^*, \varepsilon)$ the sequence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ generated by Newton's method satisfies the conditions of Theorem 4.3, and hence will converge quadratically, we must compute $\varepsilon = \min\{r, \frac{1}{2\beta\gamma}\}$ as established in (4.1). From earlier calculations we know that $r = 0.67994$ and by substituting in $\beta = 0.072$ and $\lambda = 95.20320$ from above, we obtain that $\frac{1}{2\beta\gamma} = 0.07294$. Thus ε is the minimum of the two values, $\varepsilon = 0.07294$. Theorem 4.3 guarantees that inside this neighborhood, $N(\mathbf{x}^*, 0.07294)$, the sequence of points generated by Newton's method converges quadratically to $\mathbf{x}^*$. This information is represented in Figure 4.1.3 with the inner circles representing the epsilon neighborhoods.

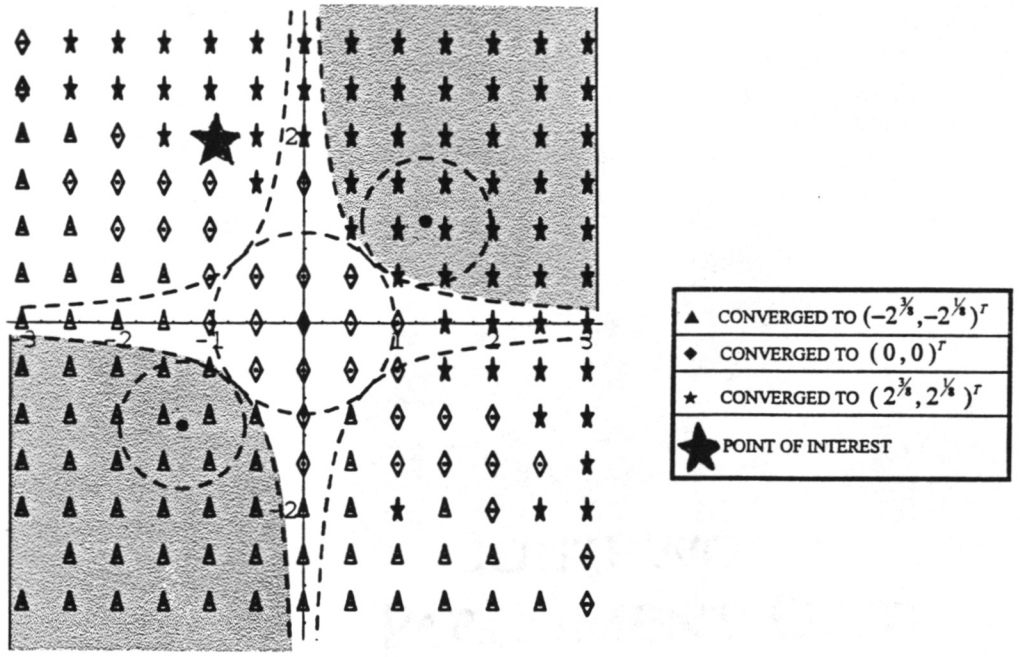
Figure 4.1.3 Epsilon neighborhoods surrounding the minimizers



A measure of relative nonlinearity of the function f at $\mathbf{x}^* = (2^{3/8}, 2^{1/8})^T$ is found by computing $\gamma_{rel} = \beta\gamma$. Again substituting in the previously calculated values for β and γ we obtain that $\gamma_{rel} = 6.85463$.

Figure 4.1.4 represents the coordinate points at which Newton's method was started and the critical points to which they converged using the *findRoot* function for which a definition and code are given in the Appendix. One interesting thing to note is that Newton's method did not necessary send a point to the nearest critical point which may seem contradictory at first glance since the logic behind the theorem is finding a "good enough" starting point from which to begin Newton's method. This finding reinforces the fact that proximity to the critical point is not the only consideration necessary to determine a "good" starting point for Newton's method although that is enough if inside the ε-neighborhood. Look in particular at the points $(1,-2)^T$ and $(-1,2)^T$ which are closer to either of the other critical points but actually converge to the critical point that is farthest away.

Figure 4.1.4 Convergence Patterns



For all of the algorithms applied to this function (i.e., the Newton methods FindRoot and *findRoot* and the steepest descent methods FindMinimum and *gradsearch*) the point $(-1, 2)^T$ converged to a critical point but not always the same critical point. We have chosen this point to illustrate the different paths taken by the iterates of several of the algorithms being compared. To help the reader distinguish between user defined functions and *Mathematica's* built-in functions we have italicized user defined functions and underlined *Mathematica's* functions. Table 4.1.3 lists each algorithm and gives a brief explanation of the method used.

Table 4.1.3 Description of algorithms being compared

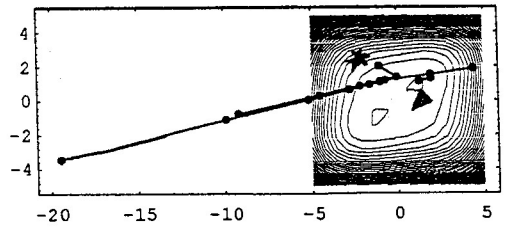
Algorithm name	Optimization algorithm incorporated
1. FindRoot - a built-in *Mathematica* function	Uses a modified Newton's method to find the local root of a given function. For minimization, this program is applied to the gradient.
2. *findRoot* – a user defined program	Uses a pure Newton's method to find the local root of a given function.
3. FindMinimum – a built-in *Mathematica* function	Uses a modified steepest descent method to find the minimizer of a multivariable function. This program is applied to the function itself (not the gradient).
4. *gradsearch* – a user defined program	Uses a pure steepest descent method in order to find the minimizer of a multivariable function. This program is also applied to the function itself and not the gradient.

All of these algorithms evaluate the gradient in some way so as to determine if the results are zero within some tolerance. For the user-defined algorithms an l_∞ norm is used while the norms used with *Mathematica's* built-in functions are not known. With some manipulation we were able to obtain the iterate paths for all of the programs with the exception of the FindMinimum algorithm and the number of iterations required to satisfy the stopping criterion. In Figure 4.1.5 these paths have been traced beginning from the point $(-1, 2)^T$.

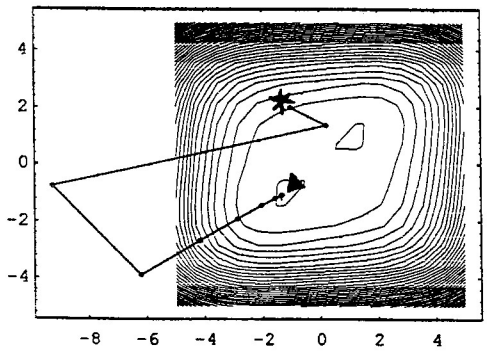
Figure 4.1.5 Iterate path for each algorithm when started at the test point $(-1,2)^T$

✳	STARTING POINT
▲	ENDING POINT

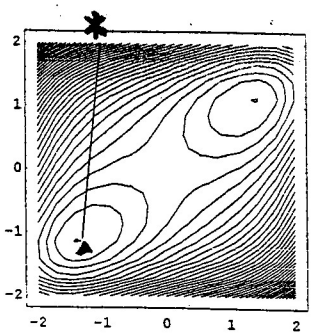
FindRoot results



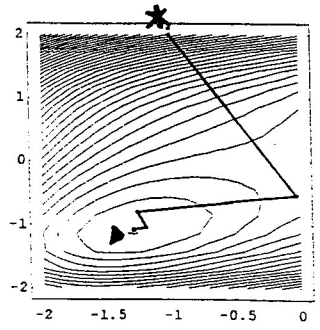
***findRoot* results**



FindMinimum results



***gradsearch* results**



In comparing these methods, recall all converged to a critical point but not all to the same critical point or at a similar pace. Table 4.1.4 lists the number of iterations required by each method to satisfy its stopping criterion and the point at which the algorithm was satisfied. Recall that not all algorithms are necessarily using the same stopping criterion; therefore no quantitative choice of "best" can be made from these comparisons.

Table 4.1.4 Comparison of algorithms / iteration count

Algorithm	Number of iterations required to satisfy the stopping criterion	Point at which the algorithm's stopping criterion were satisfied
FindRoot	26	$(2^{3/8}, 2^{1/8})^T$
findRoot	11	$(-2^{3/8}, -2^{1/8})^T$
FindMinimum	1	$(-2^{3/8}, -2^{1/8})^T$
gradsearch	17	$(-2^{3/8}, -2^{1/8})^T$

As listed earlier both FindRoot and *findRoot* use Newton's method, but they take drastically different paths and to different critical points. For these algorithms we found that *findRoot* was taking a pure Newton step while the FindRoot algorithm was altering this step in some way. For this test point the two methods agreed on the first two iterations then differ drastically on the third iteration with FindRoot being sent off to the point $(-19.5332,-3.4487)^T$ (see Figure 4.1.5). It then continues until the stopping criterion is satisfied, which occurs at a critical point different from the pure Newton method *findRoot.*

The differing results for these algorithms can easily be seen with the single variable function $f(x) = x^{1/3}$ which was given in [8, p.171] as an example in which FindRoot failed to converge to the prescribed accuracy. Applying the

pure Newton method gives solutions that alternate signs and in the pattern $x^{(k)} = -2x^{(k-1)}$. Table 4.1.5 shows that *findRoot* did in fact behave as expected while FindRoot produced markedly different results.

Table 4.1.5 Comparison of the single-variable FindRoot and *findRoot* algorithms for $f(x) = x^{1/3}$ (starting point at $x = 0.1$)

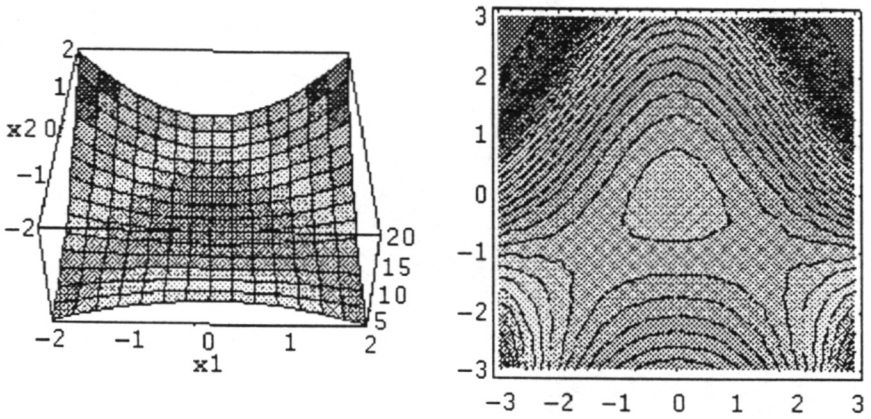
Iteration count	FindRoot Results	*findRoot* Results
1	-0.2	-0.2
2	-0.05	0.4
3	-0.05	-0.8
4	-0.05	1.6
5	-0.0125	-3.2
6	-0.0125	6.4
7	-0.0125	-12.8
8	-0.003125	25.6
9	-0.003125	-51.2
10	-0.003125	102.4

These results bring out an interesting point about using pre-existing programs for optimization. It is obviously cost-efficient to use a pre-existing program but it is also necessary to understand the adaptations made to the pure algorithms in order to predict the behavior of the "enhanced" algorithm. The need for several optimization algorithms can easily be seen in terms of cost of iterations alone. For example, if, for the algorithm involved, function evaluation is relatively inexpensive then choosing a program that requires more iterations is still feasible. However if the cost of function evaluations were high, one would want to choose the most efficient program available in terms of iteration.

Example 2 :

In Example 2 we are examining the function $f(x_1, x_2) = x_1^2 + x_2^2 + x_1^2 x_2 + 4$, which is represented in the 3-dimensional graph and contour plot below.

Figure 4.2.1 $f(x_1, x_2) = x_1^2 + x_2^2 + x_1^2 x_2 + 4$



The critical points of this function are $(\sqrt{2},-1)^T$, $(-\sqrt{2},-1)^T$ and $(0,0)^T$. The Hessian matrix for this function is $\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} 2+2x_2 & 2x_1 \\ 2x_1 & 2 \end{bmatrix}$

and we have determined at which critical points the Hessian is positive definite which is a sufficient condition for the point to be a minimizer of f.

Table 4.2.1

Critical Point	Is the Hessian positive definite at this point?	Is the Hessian singular at this point?	Minimizer, Maximizer or Saddle point?
1. $(0,0)^T$	Yes	No	Minimizer
2. $(\sqrt{2},-1)^T$	No	No	Saddle point
3. $(-\sqrt{2},-1)^T$	No	No	Saddle point

The Hessian of this function is nonsingular on the set $\{\mathbf{x} | x_2 \neq x_1^2 - 1\}$.

In order to determine P_1 surrounding the first critical point we must use the information from the analysis of the Hessian matrix and where it is positive

definite. For example 2 these conditions are: 1.) $x_2 > -1$ and 2.) $x_2 > x_1^2 - 1$. The region where the Hessian is positive definite is the intersection of the regions resulting from these conditions which is $x_2 > x_1^2 - 1$. The minimizer of this function is the point $(0, 0)^T$. The Hessian matrix when evaluated at $\mathbf{x}^* = (0,0)^T$ is

$$\mathbf{H}_f(\mathbf{x}^*) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \text{ and } \mathbf{H}_f(\mathbf{x}^*)^{-1} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}.$$

To calculate the value of β we have the following inequality:

$$\| \mathbf{H}_f(\mathbf{x}^*)^{-1} \|_2 \leq \beta .$$

The spectral radius of this matrix is 0.5 and therefore $0.5 \leq \beta$.

To find the largest open neighborhood containing $\mathbf{x}^*$ and on which the Hessian is nonsingular we need the minimum distance from the boundary of the region where the Hessian is nonsingular and the function minimizer, which is represented by the following function:

$$g(x_1) = (x_1 - 0)^2 + (x_1^2 - 1 - 0)^2 .$$

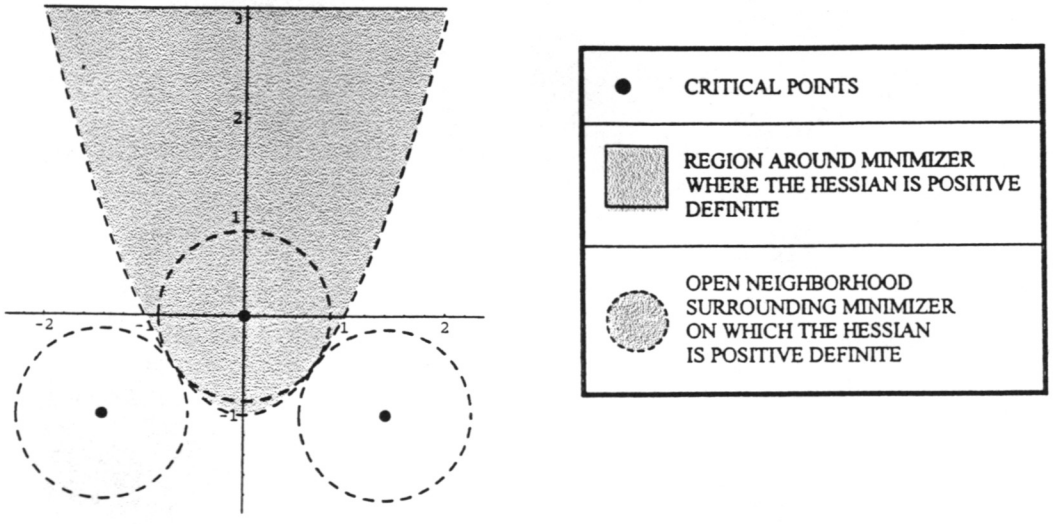
The minimizer of g is at the point $(\sqrt{2}, 8.48528)^T$, which lies on the boundary of this region. The distance between $(0, 0)^T$ and $(\sqrt{2}, 8.48528)^T$ is 0.86023 which will be used as the radius of the open neighborhood. Thus the largest open neighborhood containing $\mathbf{x}^*$ and contained in P_1 is $N(\mathbf{x}^*, 0.86023)$. Similar calculations find the largest open neighborhoods surrounding each of the critical points on which the Hessian is nonsingular. (Recall that the Hessian is not positive definite at the other critical points and therefore no such P_m exist for these points.) Table 4.2.2 lists the values of these neighborhoods for each critical point if they exist.

Table 4.2.2

Critical point	Largest open neighborhood surrounding the critical point on which $\mathbf{H}_f(\mathbf{x})$ is nonsingular	P_m = largest open convex set containing the critical point such that $\mathbf{H}_f(\mathbf{x})$ is positive definite
1. $(0,0)^T$	$N((0,0)^T, 0.86023)$	$P_1 = \{\mathbf{x} \mid \det \mathbf{H}_f(\mathbf{x}) > 0\}$
2. $(\sqrt{2},-1)^T$	$N((\sqrt{2},-1)^T, 0.86023)$	Does not exist-Hessian is not positive definite at this point
3. $(-\sqrt{2},-1)^T$	$N((-\sqrt{2},-1)^T, 0.86023)$	Does not exist-Hessian is not positive definite at this point

Figure 4.2.2 shows the critical points for this function, the region surrounding the minimizer where the Hessian is positive definite, and the largest open neighborhood surrounding each critical point such that the conditions for the r-neighborhood are satisfied.

Figure 4.2.2 Region where the Hessian is positive definite



In order to determine the region where quadratic convergence is guaranteed by the theorem and the relative nonlinearity of the function f we will need a Lipschitz constant γ such that $\mathbf{H}_f \in Lip_\gamma(\mathrm{N}(\mathbf{x}^*,r))$, i.e.

$\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \|_2 \le \gamma \| \mathbf{x} - \mathbf{y} \|_2$ for all $\mathbf{x},\mathbf{y} \in \mathrm{N}(\mathbf{x}^*,r))$ with $\mathbf{x}^* = (0,0)^T$.

Using the Hessian we obtain

$$\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) = \begin{bmatrix} 2+2x_2 & 2x_1 \\ 2x_1 & 2 \end{bmatrix} - \begin{bmatrix} 2+2y_2 & 2y_1 \\ 2y_1 & 2 \end{bmatrix} = \begin{bmatrix} 2(x_2-y_2) & 2(x_1-y_1) \\ 2(x_1-y_1) & 0 \end{bmatrix}$$

Calculating with the infinity norm gives

$$\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \|_\infty = Max\{ 2|x_2-y_2| + 2|x_1-y_1|, 2|x_1-y_1| \}$$

$$= 2\{ |x_2-y_2| + |x_1-y_1| \}.$$

Using (2.4) this implies

$$\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \|_\infty \le 4 \| \mathbf{x} - \mathbf{y} \|_\infty .$$

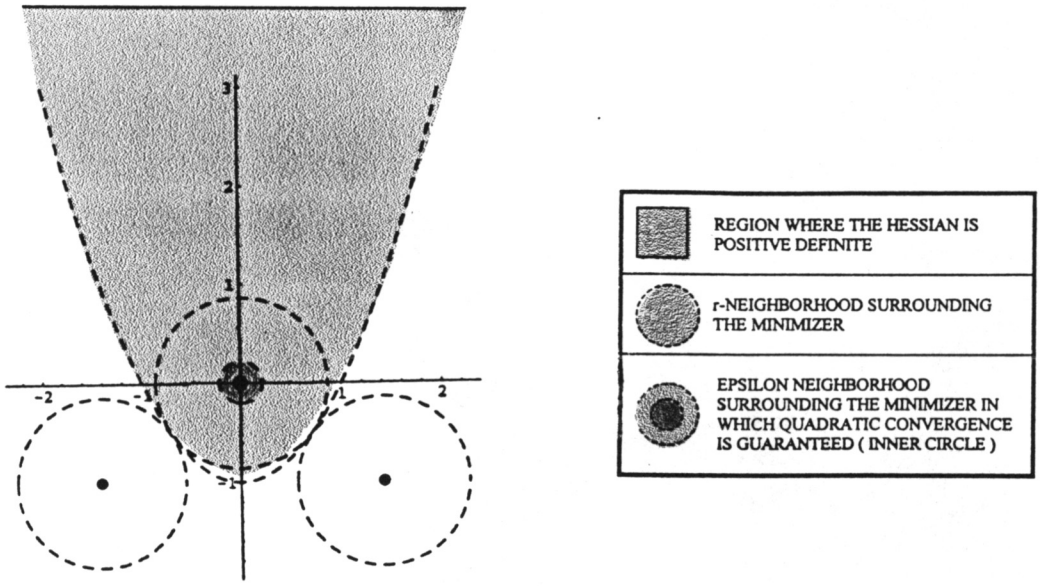
Using (4.2) and (4.3), a value for gamma in terms of the two-norm can be found yielding

$$\| \mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) \|_2 \le 5.65685 \| \mathbf{x} - \mathbf{y} \|_2 \text{ and therefore } \gamma = 5.65685 .$$

To find the neighborhood $\mathrm{N}(\mathbf{x}^*,\varepsilon)$ such that for any $\mathbf{x}^{(0)} \in \mathrm{N}(\mathbf{x}^*,\varepsilon)$ the sequence $\mathbf{x}^{(1)},\mathbf{x}^{(2)},\ldots$ generated by Newton's method converges quadratically to $\mathbf{x}^*$ the quantity $\varepsilon = \min\{r, \frac{1}{2\beta\gamma}\}$ is computed. From earlier calculations we know that $r = 0.86023$ and by substituting $\beta = 0.5$ and $\lambda = 5.65685$ from above, we obtain $\frac{1}{2\beta\gamma} = 0.17678$ and thus $\varepsilon = 0.17678$. Hence an open neighborhood such that the sequence of points generated by Newton's method is guaranteed to converge quadratically to $\mathbf{x}^*$ is $\mathrm{N}(\mathbf{x}^*,0.17678)$. This information is

represented in Figure 4.2.3 with the inner circle representing the epsilon neighborhood.

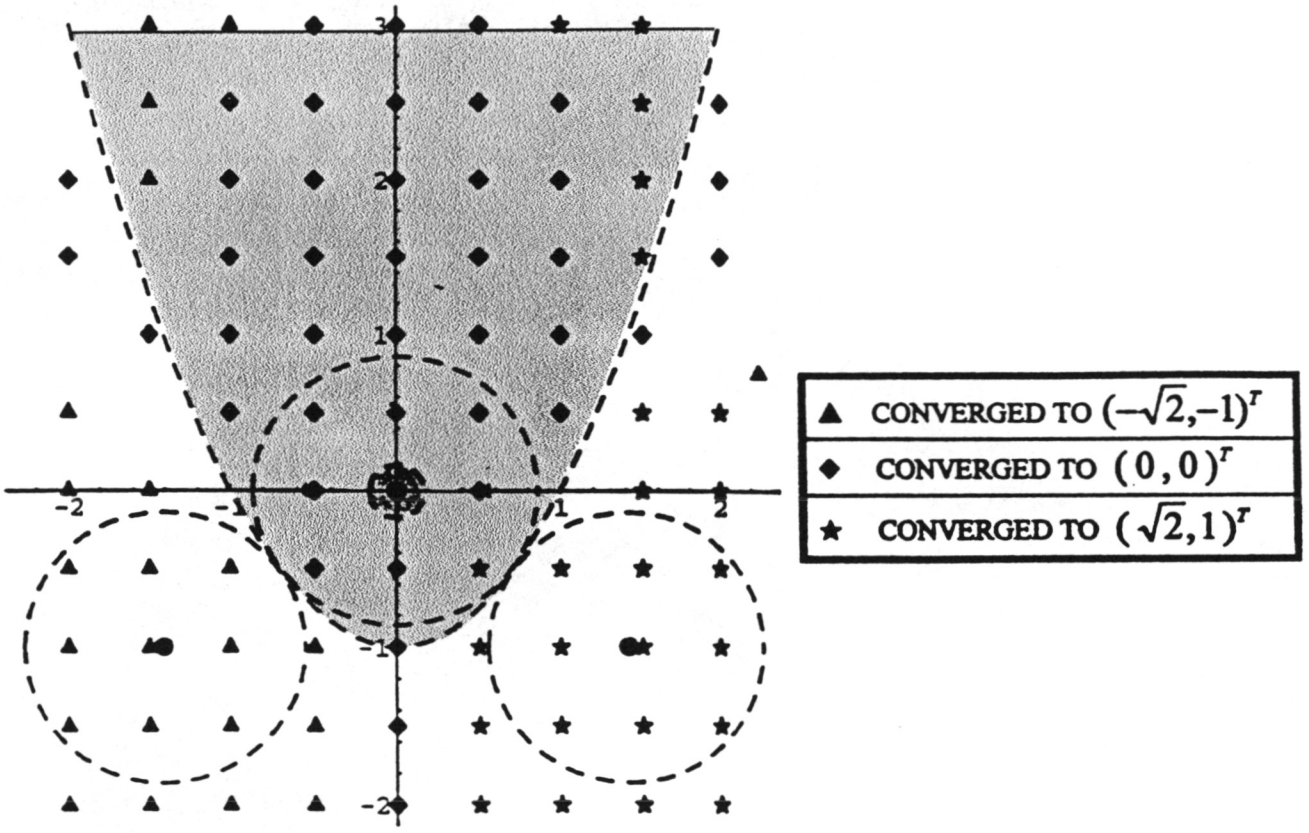
Figure 4.2.3 Epsilon neighborhood surrounding the minimizer



Again it should be noted that this is a worst case estimate for quadratic convergence. Substituting in the previous calculated values for β and γ we obtain that $\gamma_{rel} = 2.82843$.

Figure 4.2.4 shows the results of Newton's method applied to a grid in the coordinate plane. Each point was used to begin the *findRoot* function (see Appendix) and the results were graphed. Also included on the graph are the largest open neighborhood surrounding the minimizer in which the Hessian is positive definite and the ε-neighborhood in which we can expect quadratic convergence. Note that convergence to the minimizer does occur outside of this ε-neighborhood, but no guarantees concerning the rate of convergence are provided by the theorem outside of this region.

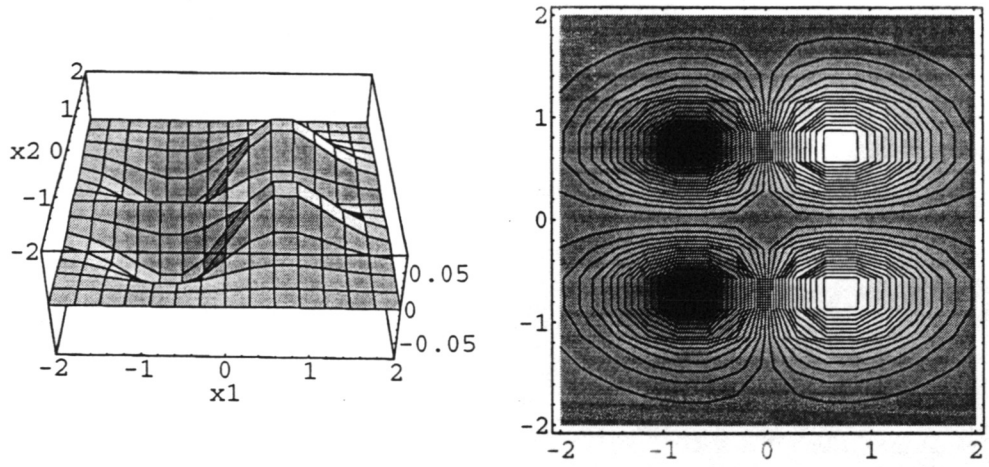
Figure 4.2.4 Convergence patterns



Example 3:

In Example 3, we will consider a function with an exponential factor, $f(x_1,x_2)=x_1x_2^2e^{(-x_1^2-2x_2^2)}$. One interesting feature of this function is its infinite number of critical points. The following graphs are a 3-dimensional representation of the function and its two dimensional contour plot.

Figure 4.3.1 $f(x_1, x_2) = x_1 x_2^2 e^{(-x_1^2 - 2x_2^2)}$



The critical points of this function are $(\sqrt{2}/2, \sqrt{2}/2)^T$, $(\sqrt{2}/2, -\sqrt{2}/2)^T$, $(-\sqrt{2}/2, \sqrt{2}/2)^T$, $(-\sqrt{2}/2, -\sqrt{2}/2)^T$ and the entire x_1-axis. The Hessian matrix for this function is

$$\mathbf{H}_f(\mathbf{x}) = 2e^{(-x_1^2 - 2x_2^2)} \begin{bmatrix} x_1 x_2^2(-3 + 2x_1^2) & x_2(-1 + 2x_1^2)(-1 + 2x_2^2) \\ x_2(-1 + 2x_1^2)(-1 + 2x_2^2) & x_1(1 - 10x_2^2 + 8x_2^4) \end{bmatrix}.$$

Table 4.3.1 lists the maximizers, minimizers and saddle points for this function and at which critical points the Hessian is positive definite.

Table 4.3.1

Critical Point	Is the Hessian positive definite at this point?	Is the Hessian singular at this point?	Minimizer, maximizer or a saddle point?
1. $(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T$	No	No	Maximizer
2. $(\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T$	No	No	Maximizer
3. $(-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T$	Yes	No	Minimizer
4. $(-\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T$	Yes	No	Minimizer
5. Along the entire x_1-axis	No- not at any of these points	No	Saddle points

The Hessian of f is nonsingular whenever $-1+x_1^2-2x_1^4+4x_2^2+14x_1^2x_2^2-4x_1^4x_2^2-4x_2^4-8x_1^2x_2^4\neq 0$. The regions in which the Hessian matrix of this function is positive definite are represented by the shaded regions in Figure 4.3.2.

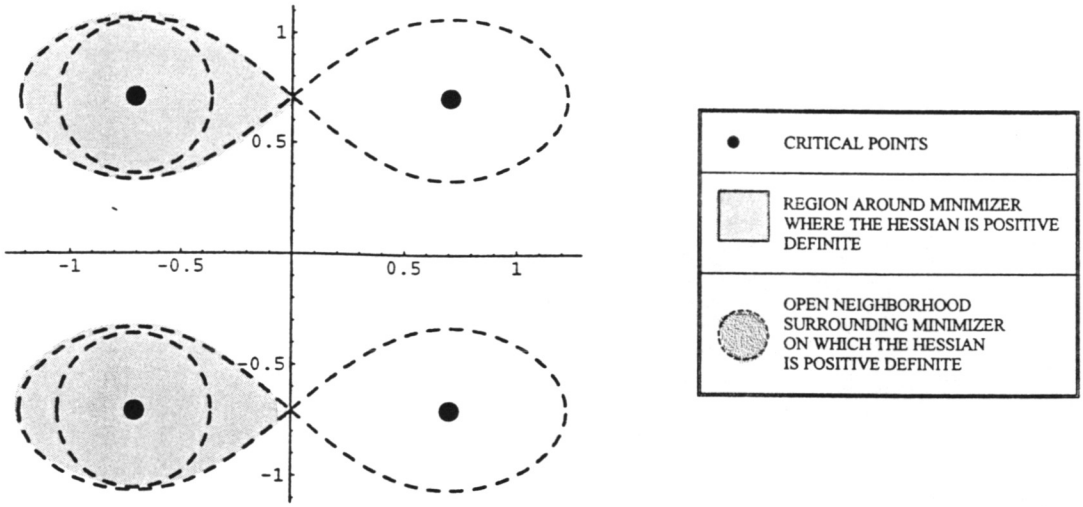
To find a largest open neighborhood containing the third critical point, $\mathbf{x}^* = (-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T$ and contained in P_3 we need the minimum distance from the boundary of the region where the Hessian is positive definite and the minimizer itself. Because of the difficulty in determining this function explicitly in terms of x_1 and x_2 and estimate for the radius will be used, $r = 0.348$ (which was obtained by using graphic representations of approximations for r and choosing a value that was visibly inside the region required). Thus one of the largest open neighborhoods containing $\mathbf{x}^*$ and contained in P_3 is $N((-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T, 0.348)$. Because of the symmetries exhibited by this function the same size radius is appropriate for the second minimizer of this function which is at $(-\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T$. In the following table we have listed each critical point, the largest region where the Hessian is positive definite and contains the critical point and our estimate of the largest open neighborhood surrounding each of the critical points on which the Hessian is nonsingular.

Table 4.3.2

Critical point	Largest open neighborhood surrounding the critical point on which $\mathbf{H}_f(\mathbf{x})$ is nonsingular	$\mathbf{P}_m$ = largest open convex set containing the critical point such that $\mathbf{H}_f(\mathbf{x})$ is positive definite
1. $(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T$	$N((\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T,0.348)$	Does not exist—Hessian is not positive definite at this point
2. $(\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T$	$N((\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T,0.348)$	Does not exist—Hessian is not positive definite at this point
3. $(-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T$	$N((-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})^T,0.348)$	$\mathbf{P}_3 = \{\mathbf{x} \mid \det \mathbf{H}_f(\mathbf{x}) > 0, x_1 < 0, x_2 > 0\}$
4. $(-\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T$	$N((-\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2})^T,0.348)$	$\mathbf{P}_4 = \{\mathbf{x} \mid \det \mathbf{H}_f(\mathbf{x}) > 0, x_1 < 0, x_2 < 0\}$
5. the entire x-axis	varies with each critical point	Does not exist—Hessian is not positive definite at any of these points

This information is represented graphically in Figure 4.3.2.

Figure 4.3.2 Regions where the Hessian is positive definite



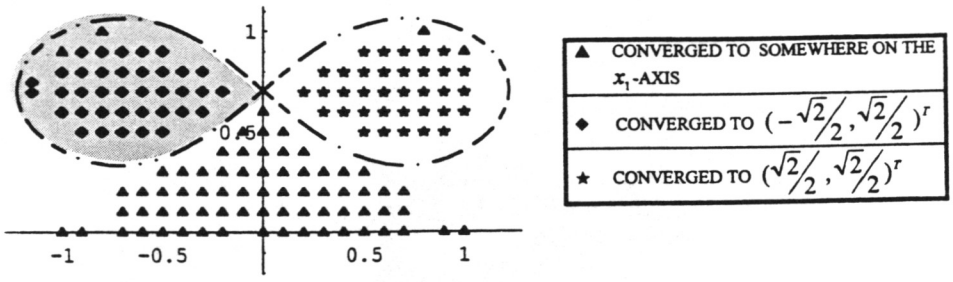
While it is possible to calculate an epsilon neighborhood on which quadratic convergence is guaranteed using the Mean Value Theorem for functions of two variables ([1, p.809], for example), the calculation is tedious and the result obtained is too small to be clearly visible and hence will not be presented.

Figure 4.3.3 shows the points on the grid (where $x_1 \in [-2,2]$ and $x_2 \in [-2,2]$) which did converge to a critical point of this function. Recall that this function has an infinite number of critical points along the x_1 axis, none of which is a minimizer of the function. In order to reach the goal of minimization it would be particularly important to determine the regions where

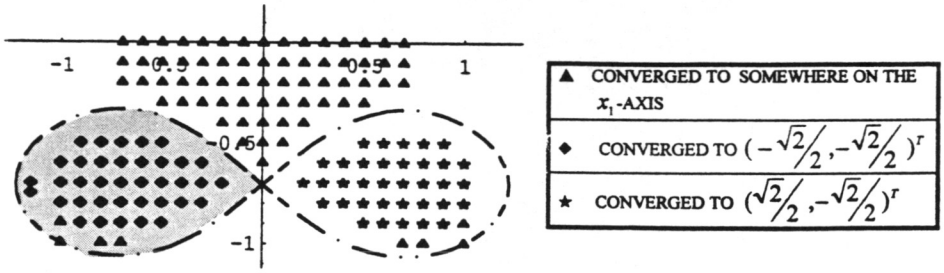
the Hessian was positive definite and avoid the regions where the minimizers could not occur.

Figure 4.3.3 Convergence patterns

Upper plane



Lower plane



For six points on this grid the Newton's algorithms gave a warning message concerning a badly conditioned matrix or an answer was given that did not correspond with a critical point for this function. For four of these points that is to be expected because the Hessian is singular at those points. These points are $(-1,-1)^T, (-1,1)^T, (1,-1)^T$ and $(1,1)^T$. At two other points, $(0,1)^T$ and $(0,-1)^T$, the Newton algorithms stopped prematurely at a point other than a critical point of the function, which means that the stopping

criteria for these algorithms were satisfied although a critical point had not been reached. Figure 4.3.4 shows the three-dimensional graph of the function and the contour plot with the figure-eight shapes representing the points where the Hessian is singular with the points of interest enlarged. Notice the symmetry exhibited by this function not only with the critical points but also the points we are interested in.

Figure 4.3.4 Points of Interest

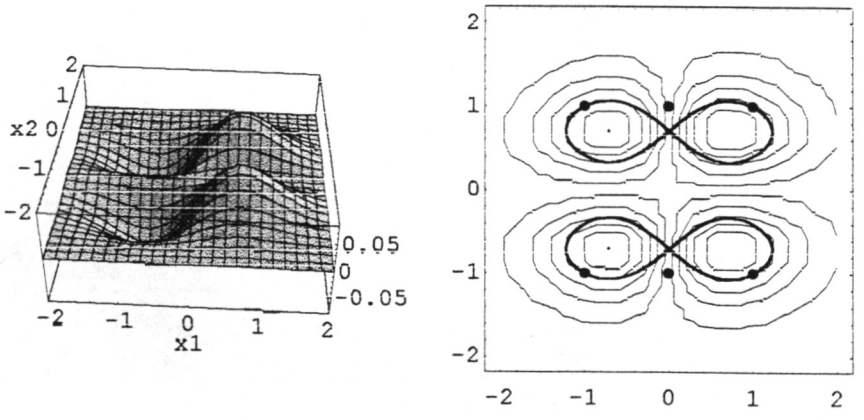


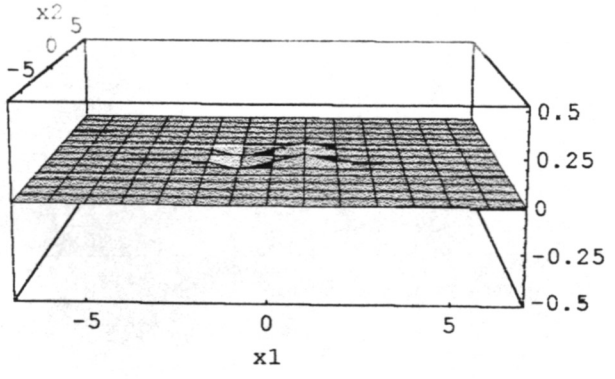
Table 4.3.3 lists the points of interest and how each of the algorithms behaved when started at these points. Recall that the stopping criterion is not necessarily the same for all algorithms.

Table 4.3.3 Points of interest and how each algorithm behaved when started at these points

Point of Interest	Newton Algorithms		Steepest Descent Algorithms	
	findRoot	**FindRoot**	***gradsearch***	**FindMinimum**
$(-1,-1)^T$	Badly conditioned matrix error message	Badly conditioned matrix error message	Converged to $(-\sqrt{2}/2,-\sqrt{2}/2)^T$ in 13 iterations	Converged to $(-\sqrt{2}/2,-\sqrt{2}/2)^T$ in 1 iteration
$(-1,1)^T$	Badly conditioned matrix error message	Badly conditioned matrix error message	Converged to $(-\sqrt{2}/2,\sqrt{2}/2)^T$ in 13 iterations	Converged to $(-\sqrt{2}/2,\sqrt{2}/2)^T$ in 1 iteration
$(1,-1)^T$	Badly conditioned matrix error message	Badly conditioned matrix error message	Error message stating that the minimizer could not be bracketed in 250 iterations	Vanishing gradient error message
$(1,1)^T$	Badly conditioned matrix error message	Badly conditioned matrix error message	Error message stating that the minimizer could not be bracketed in 250 iterations	Vanishing gradient error message
$(0,1)^T$	Stopped in 16 iterations at $(0,3.298549)^T$ which is not a critical point	Stopped in 15 iterations at $(0,2.94154)^T$ which is not at a critical point	Converged to $(-\sqrt{2}/2,\sqrt{2}/2)^T$ in 3 iterations	Converged to $(-\sqrt{2}/2,\sqrt{2}/2)^T$ in 1 iteration
$(0,-1)^T$	Stopped in 16 iterations at $(0,-3.28549)^T$ which is not a critical point	Stopped in 15 iterations at $(0,-2.94154)^T$ which is not at a critical point	Converged to $(-\sqrt{2}/2,-\sqrt{2}/2)^T$ in 3 iterations	Converged to $(-\sqrt{2}/2,-\sqrt{2}/2)^T$ in 1 iteration

The Newton algorithms obviously had trouble at the points for which the Hessian is singular as expected. For the remaining two points, $(0,1)^T$ and $(0,-1)^T$, the stopping criteria for both Newton algorithms allowed the algorithms to stop prematurely at points that were not critical points of the function. Recall that the stopping criteria for the user-defined algorithms require the components of the gradient to be less than a given tolerance. In order to see why these algorithms claimed to be successful at these points, it will be helpful to look at this function on a slightly larger domain than was previously shown.

Figure 4.3.5 Function shown on a broader domain



Notice that the function is extremely flat at the points for which the stopping criteria were satisfied. This finding emphasizes the need to choose an appropriate stopping criterion that will allow the algorithm to continue until success is achieved but at the same time not allow the algorithm to continue needlessly when it is "close" enough to the appropriate solution. It also points

out some of the limitations of Newton's method, i.e., how it can "stall" out in an area that is extremely flat (at which the gradient is approximately zero) and still not be at a critical point of the function.

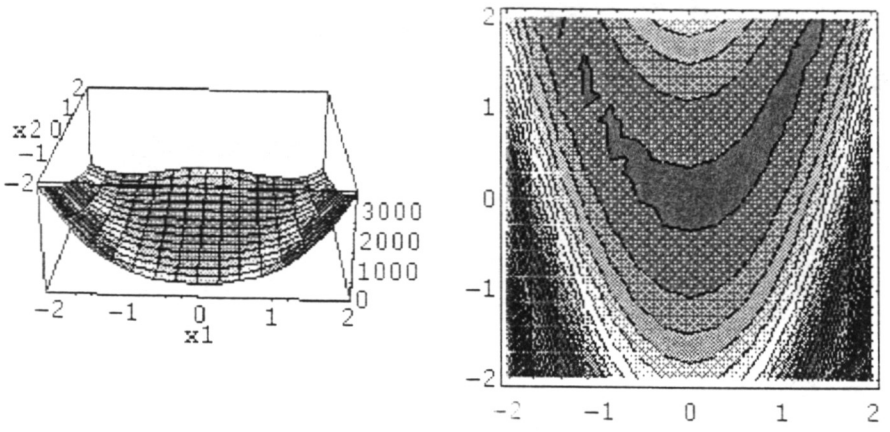
The FindMinimum and *gradsearch* algorithms were successful at four of the six points of interest. The steepest descent algorithm incorporated by both of these algorithms is a first-order method and requires only the calculation of the first order partial derivatives; therefore it was not hindered by whether or not the Hessian was singular at these points. In doing the calculations using the pure steepest descent method we found that the *gradsearch* algorithm was taking steps consistent with the pure algorithm while the FindMinimum algorithm is not; therefore the FindMinimum algorithm must be enhanced in some way in order to reach the minimizer in only one iteration. FindMinimum was successful at two of the points where the Hessian is singular and gave an error message concerning a vanishing gradient at the other test points where the Hessian is singular.

These findings point to more considerations in choosing an optimization program. Determining whether to use a first or second order method may be limited by the tractability of calculating the partial derivatives. In practice however, partial derivatives are most likely not calculated explicitly but a substitution with a secant approximation is often made.

Example 4:

In example 4 we are considering the function $f(x_1, x_2) = (1 - x_1)^2 + 100(x_1^2 - x_2)^2$ which is represented in Figure 4.4.1. This function is the Rosenbrock "banana" function and is commonly used as a test problem for minimization algorithms [7,9].

Figure 4.4.1 Rosenbrock's Banana Function



This function has a unique critical point $(1,1)^T$. The Hessian matrix for this function is

$$\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} 2+800x_1^2+400(x_1^2-x_2) & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

and it is positive definite when evaluated at the critical point, which guarantees that the point $(1,1)^T$ is a minimizer. (See Table 4.4.1).

Table 4.4.1

Critical point	Is the Hessian positive definite at this point?	Is the Hessian singular at this point?	Minimizer, Maximizer or Saddle point?
1. $(1,1)^T$	Yes	No	Minimizer

The Hessian is nonsingular on the set $\{\mathbf{x} | x_2 \neq x_1^2 + 0.005\}$. Using the information from the analysis of Hessian matrix we must determine P_1 for the critical point. For this particular function the Hessian is positive definite when the following conditions are satisfied: 1.) $2+800x_1^2+400(x_1^2-x_2)>0$ and 2.) $400+80000x_1^2-80000x_2>0$. Since the first condition is satisfied when

$x_2 < 3x_1^2 + 0.005$ and the second condition is satisfied when $x_2 < x_1^2 + 0.005$, the region where the Hessian is positive definite is the intersection of these two regions i.e. where $x_2 < x_1^2 + 0.005$. The minimizer for the Rosenbrock function is $(1,1)^T$. The Hessian matrix, when evaluated at $\mathbf{x}^* = (1,1)^T$ is

$$\mathbf{H}_f(\mathbf{x}^*) = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix} \text{ and } \mathbf{H}_f(\mathbf{x}^*)^{-1} = \begin{bmatrix} 0.05 & 1 \\ 1 & 2.005 \end{bmatrix}.$$

Using the relation $\| \mathbf{H}_f(\mathbf{x}^*)^{-1} \|_2 \leq \beta$ we obtain $2.42589 \leq \beta$.

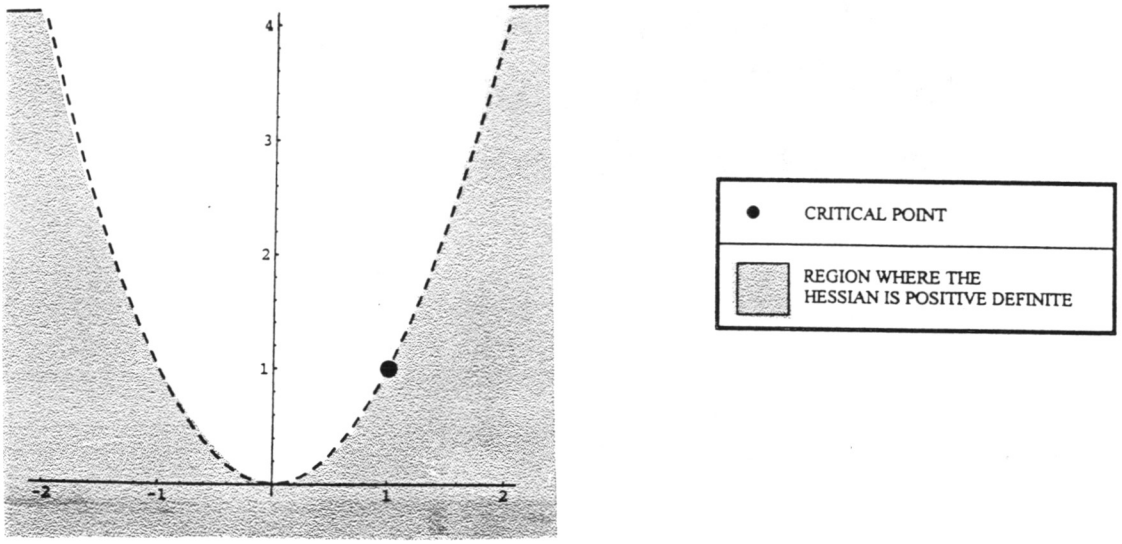
The minimum distance from the boundary of the region where the Hessian is positive definite and the minimizer $(1,1)^T$ is found by minimizing the function

$$g(x_1) = (x_1 - 1)^2 + (x_1^2 + 0.005 - 1)^2.$$

The minimizer of g occurs at the point $(0.99666, 0.99832)^T$, which lies on the boundary of the region where $\mathbf{H}_f(\mathbf{x})$ is nonsingular. The distance between $(1,1)^T$ and $(0.99666, 0.99832)^T$ is 0.00374 and will be used as the radius of the open neighborhood. Thus the largest open neighborhood containing $\mathbf{x}^*$ and contained in P is $N(\mathbf{x}^*, 0.00374)$.

The following graph shows the region where the Hessian is positive definite and the critical point. It should be noted that the open neighborhood around the minimizer is so small that it cannot be represented on the graph.

Figure 4.4.2 Region of positive definiteness



In order to determine a region in which quadratic convergence is guaranteed for the Rosenbrock function and the relative nonlinearity of this function we will need a Lipschitz constant γ such that $\mathbf{H}_f \in Lip_\gamma(\mathrm{N}(\mathbf{x}^*,r))$, i.e.

$\|\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})\|_2 \le \gamma\|\mathbf{x}-\mathbf{y}\|_2$ for all $\mathbf{x},\mathbf{y} \in \mathrm{N}(\mathbf{x}^*,r)$ with $\mathbf{x}^* = (1,1)^T$.

Using the values of $\mathbf{H}_f(\mathbf{x})$ and $\mathbf{H}_f(\mathbf{y})$ for this function we obtain

$$\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y}) = -400\begin{bmatrix} -3(x_1^2 - y_1^2) + (x_2 - y_2) & (x_1 - y_1) \\ (x_1 - y_1) & 0 \end{bmatrix}.$$

Using the infinity norm, we obtain

$$\begin{aligned}\|\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})\|_\infty &\le 400\, Max\left\{\left|-3(x_1^2 - y_1^2) + (x_2 - y_2)\right| + \left|(x_1 - y_1)\right|, \left|(x_1 - y_1)\right|\right\} \\ &= 400\left|-3(x_1^2 - y_1^2) + (x_2 - y_2)\right| + \left|(x_1 - y_1)\right|.\end{aligned}$$

On the first absolute value we use the triangle inequality (2.3) to obtain

$$\|\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})\|_\infty \le 400\left(3\left|x_1^2 - y_1^2\right| + \left|x_2 - y_2\right| + \left|x_1 - y_1\right|\right)$$

$$= 400\Big(3|x_1 - y_1|\,|x_1 + y_1| + |x_2 - y_2| + |x_1 - y_1|\Big).$$

The largest first coordinate one can have inside the ball with radius $r = 0.00374$ centered at the minimizer $(1,1)^T$ is $1 + r = 1.00374$. We can replace the $|x_1 + y_1|$ with twice the largest first coordinate possible which gives

$$\begin{aligned}\|\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})\|_\infty &\le 400\Big(3(2.00749)|x_1 - y_1| + |x_2 - y_2| + |x_1 - y_1|\Big)\\ &= 2808.98253|x_1 - y_1| + 400|x_2 - y_2|\\ &\le 2808.98253\{|x_1 - y_1| + |x_2 - y_2|\}.\end{aligned}$$

Using (2.4) yields

$$\begin{aligned}\|\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})\|_\infty &\le 2(2808.98253)Max\{|x_1 - y_1|, |x_2 - y_2|\}\\ &= 5617.96506\,\|\mathbf{x} - \mathbf{y}\|_\infty.\end{aligned}$$

Using (4.2) and (4.3) we obtain

$$\|\mathbf{H}_f(\mathbf{x}) - \mathbf{H}_f(\mathbf{y})\|_2 \le 7945.00238\|\mathbf{x} - \mathbf{y}\|_2 \text{ and } \gamma = 7945.00238.$$

To find the open neighborhood $N(\mathbf{x}^*, \varepsilon)$ in which for all $\mathbf{x}^{(0)} \in N(\mathbf{x}^*, \varepsilon)$ the sequence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ generated by Newton's method converges quadratically to $\mathbf{x}^*$ and satisfies Theorem 4.3 we must use (4.1) to calculate ε. From earlier calculations we know that $r = 0.00374$ and by substituting in $\beta = 2.42589$ and $\gamma = 7945.00237$, we obtain that $\dfrac{1}{2\beta\gamma} = 0.00003$. Thus $\varepsilon = 0.00003$ and the open neighborhood $N((1,1)^T, 0.00003)$ in which quadratic convergence is guaranteed

is an extremely small neighborhood. A much larger region of convergence was found for this function. In fact convergence occurred from any point at which the *findRoot* algorithm was begun. A measure of relative nonlinearity for this function at $\mathbf{x}^*$ is $\gamma_{rel} = 19273.70181$.

While using the Newton algorithms on this function several discrepancies were noted. The FindRoot function sent very few coordinate points to the appropriate critical point and in most cases seemed to stop in neighborhoods nowhere near the extremum. When sent through our *findRoot* we found that every point in the grid converged to the minimizer and in seven or fewer iterations. In order to determine what might cause the algorithms to declare 'convergence' and therefore stop the iterative process we looked at the stopping criteria employed by both algorithms.

Mathematica's FindRoot function uses an altered Newton step at each iteration and uses a function evaluation (using the gradient components) as its stopping criterion [15]. (The author states the stopping criterion for the single variable case but does not state which norm they are using to evaluate their stopping criteria for multivariable functions.)

For our *findRoot* function a pure Newton step is taken at each iteration and an l_∞ stopping criterion is used. In other words when the maximum component of the gradient was less than a user-supplied tolerance *tol,* the algorithm declares convergence and lists the current iterate when

$$Max\left\{\left|(\nabla f)_1\right|,\left|(\nabla f)_2\right|\right\} \le tol\,.$$

For the polynomial examples a tolerance of $tol = 10^{-8}$ was used while that same tolerance for the exponential problem was not successful. Therefore for the exponential example the tolerance was set at $tol = 10^{-12}$.

In comparing these two algorithms and their stopping criteria we must consider how they are affected by the shape of the function and by the scaling of the variables of the function. Below is a table describing the number of extremum for each of the examples, the objective value range over a given domain and the range of the typical x_1 and x_2 values.

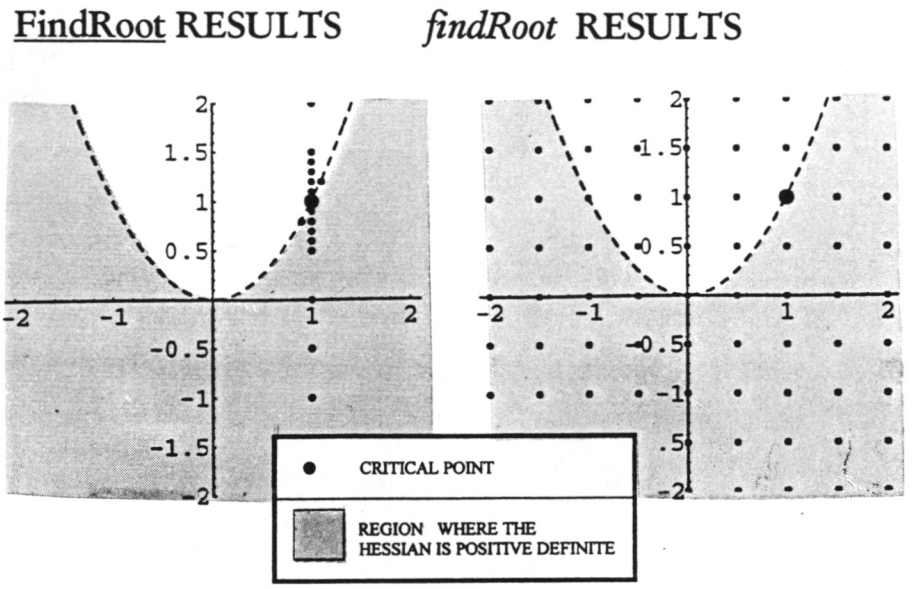
Table 4.4.2

Function	Number of Critical points	Scale for the x_1 variable	Scale for the x_2 variable	Range of function values
1. $x_1^4 + 2x_2^4 - 8x_1x_2 + 2$	2 minimizers 1 saddle point	$x_1 \in [-2,2]$	$x_2 \in [-2,2]$	$f(x_1,x_2) \in [0,80]$
2. $x_1^2 + x_2^2 + x_1^2x_2 + 4$	1 minimizer 2 saddle points	$x_1 \in [-2,2]$	$x_2 \in [-2,2]$	$f(x_1,x_2) \in [0,15]$
3. $x_1x_2^2e^{(-x_1^2-2x_2^2)}$	2 minimizers 2 maximizers infinite – saddle points	$x_1 \in [-2,2]$	$x_2 \in [-2,2]$	$f(x_1,x_2) \in [-0.05,0.05]$
4. $(1-x_1)^2 + 100(x_1^2 - x_2)^2$ Rosenbrock	1 minimizer	$x_1 \in [-2,2]$	$x_2 \in [-2,2]$	$f(x_1,x_2) \in [0,3000+]$

Notice how much more the range of the Rosenbrock function varies in contrast to the previous functions considered.

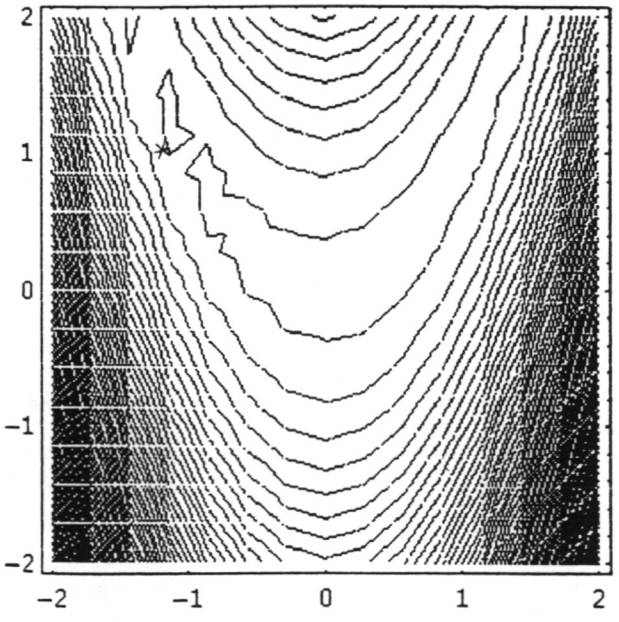
In particular we will look at the Rosenbrock function to compare the Newton and steepest descent algorithms. The figures below show the points on the grid for which *Mathematica's* FindRoot converged to the critical point and points on the grid for which our *findRoot* converged.

Figure 4.4.3



Notice in particular the point $(-1.2,1)^T$ (denoted by * in Figure 4.4.4), which lies on the opposite wall from the minimizer of this parabolic shaped valley. This point is often used as a starting point to test an optimization algorithm. To see why, it will helpful to look at a contour plot without shading shown in Figure 4.4.4.

Figure 4.4.4 Rosenbrock's Banana Function



Mathematica's FindRoot algorithm had little success in actually obtaining the unique minimizer for this function and stopped prematurely in terms of the goal which is a point where the $\nabla f \approx 0$. In fact, when the maximum number of iterations was set at 300 iterations, Mathematica's FindRoot still failed to converge within the prescribed accuracy goal stopping at $(-1.04709,1.09474)^T$ which is still on the opposite wall from the minimizer of the function. Even when the exact minimizer was given as the starting point for FindRoot, it took 4 iterations before the stopping criterion was satisfied and convergence was declared.

In *findRoot* the stopping criterion incorporates the components of the gradient and as the components of the gradient approach zero the function is getting extremely flat. This stopping criterion worked extremely well and in fact the test point $(-1.2,1)^T$, (which was on the opposite wall of the valley), converged in seven iterations with all of the coordinate points on the grid converging in at most 7 iterations.

We have chosen several test points in order to compare the paths of the Newton iterates with the steepest descent path taken by the *gradsearch* program. Table 4.4.3 compares the number of iterations taken by each method to reach the minimizer. Again, it should be noted that the stopping criterion of each algorithm is not necessarily the same , and no determination of which algorithm is "best" is implied by these comparisons.

Table 4.4.3 Points of interest and how each algorithm behaved when started at these points

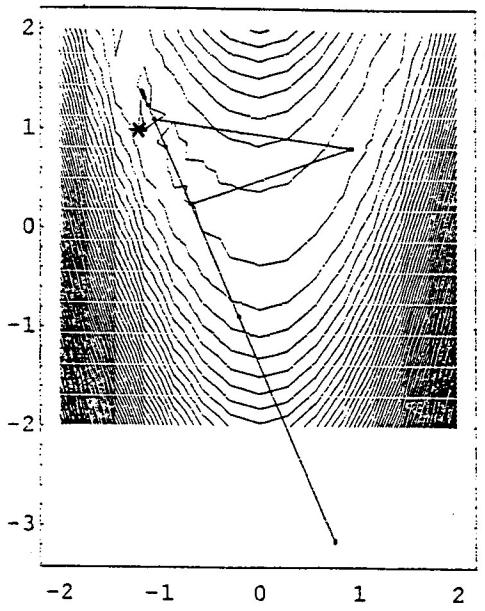
Point of interest	Newton Algorithms		Steepest Descent Algorithms	
	findRoot	FindRoot	*gradsearch*	FindMinimum
$(-1.2,1)^T$	Converged in 7 iterations	Failed to converge after 1000 iterations	Failed to converge after 1000 iterations	Converged in 214 iterations
$(-1,1)^T$	Converged in 2 iterations	Failed to converge after 1000 iterations	Converged in 1 iteration	Converged in 228 iterations
$(0,1)^T$	Converged in 5 iterations	Converged in 627 iterations	Failed to converge after 1000 iterations	Converged in 145 iterations
$(0.5,1)^T$	Converged in 5 iterations	Converged in 371 iterations	Converged in 961 iterations	Converged in 82 iterations

The paths taken by the iterates of the Newton algorithms starting at the point $(-1.2,1)^T$ are shown in Figure 4.4.5.

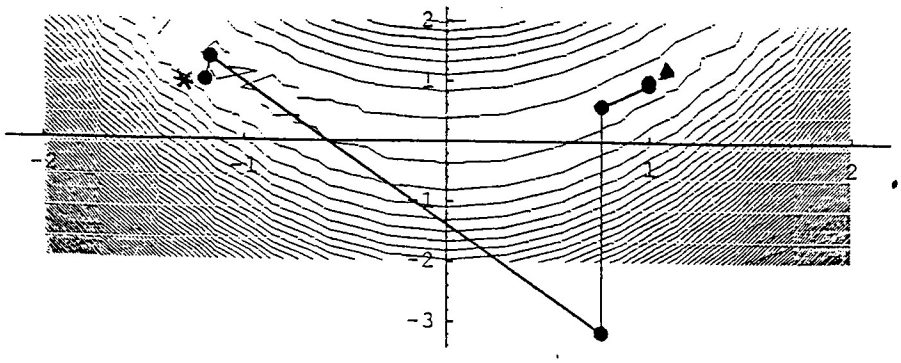
Figure 4.4.5 Iterate paths for the Newton algorithms when started at the point $(-1.2, 1)^T$

*	STARTING POINT
▲	ENDING POINT

FindRoot results



***findRoot* results**



This point clearly shows the attributes and drawbacks of the pure steepest descent method. (Recall from a previous example that *gradsearch* is using a pure steepest descent method and FindMinimum has been altered in some way which in this case allowed it to converge in fewer iterations). Within two iterations the pure steepest descent method is at a point on the opposite side of the valley and then is literally inching it's way toward the minimizer. (Recall that in over 1000 iterates this method had still not reached the minimizer.) Again this is one of the reasons this method is chosen in conjunction with Newton's method. Its global convergence properties bring the iterate into a "good" neighborhood within relatively few iterations but then settles into a slow crawl toward the minimizer. Once the iterate has made it into a "good" neighborhood, a neighborhood where Newton's method can be successful, Newton's method can begin. The local convergence properties of Newton's method then guarantee success in finding the local minimizer. Combined, these two algorithms create a globally convergent method which, once Newton's method is started, will converge quite rapidly and eventually quadratically to the minimizer of the function.

It is easily seen, even in the small number of functions examined in this thesis, that there are many significant factors that must be considered in choosing an appropriate optimization algorithm. In considering functions with only two variables, as were presented in this thesis, we noted many attributes of the given function simply because we could represent these functions graphically and examine what might be occurring at the various starting points. In most applications of optimization, however, this is simply not feasible with a typical operations research problem having hundreds of variables. But, most of the problems that were noted have a multidimensional analog that must be considered for the larger problems. And as expected, as the number of

variables in the function increases so do the considerations that must be made in choosing an optimization algorithm suitable for that function.

Throughout this thesis it has been noted that the problems of optimization and root-finding are closely related although not equivalent. In particular, for a reasonably behaved function, Newton's method was shown to be quadratically convergent when applied in a neighborhood "close" to the minimizer. It was also shown that both Newton's method in its pure form and modified versions of Newton's method are not always suitable to accomplish optimization. Although the simplicity of Newton's method (along with its rapid local convergence) adds to its attractiveness as an optimization algorithm, many more complicated algorithms have been developed that are equally successful for a variety of optimization problems. It should be noted that several of these algorithms incorporate Newton's method as well.

Although Newton's method cannot be considered as a general "all-purpose" algorithm for nonlinear optimization (such as the simplex method is for linear programming), it must be recognized for its ability to accomplish the goal of optimization either alone or in part for both constrained and unconstrained nonlinear optimization problems.

APPENDIX

The following code defines a function named "*findRoot*" (following [8]) which will find the critical points of a multivariable function using Newton's method for a given ordered pair and will list the values of the x_1 and x_2 coordinates at each iteration. The stopping criterion for this code allows it to iterate until each element of the gradient vector is less than the given epsilon which is an l_∞ stopping criterion. An example of the output generated is provided below the code.

```
f[x_,y_]=(1-x)^2+100(x^2-y)^2
funx[x_,y_]=D[f[x,y],x]
funy[x_,y_]=D[f[x,y],y]
grad[x_,y_]={{funx[x,y]},{funy[x,y]}}
funxx[x_,y_]=D[funx[x,y],x]
funyy[x_,y_]=D[funy[x,y],y]
funxy[x_,y_]=D[funx[x,y],y]
hess[x_,y_]={{funxx[x,y],funxy[x,y]},
        {funxy[x,y],funyy[x,y]}}
findRoot[f_,initx_,inity_,eps_]:=
   Module[{x=initx,y=inity,count=0},Print[x," , ",y];
          component1=N[Abs[funx[x,y]]];
          component2=N[Abs[funy[x,y]]];

       While[Max[component1,component2]>eps,
          (*Print["funx = ",component1];*)
          (*Print["   funy = ",component2];*)

                count=count+1;


            diffnew=Inverse[hess[x,y]].grad[x,y];

              x=x-N[diffnew[[1,1]]];
              y=y-N[diffnew[[2,1]]];
```

```
            component1=N[Abs[funx[x,y]]];
            component2=N[Abs[funy[x,y]]];

        (*Print["at the end of ",count," iterations,the
            gradient components are ",component1,"   ",
            component2];*)

        Print["next iterate is ","(",x,",",y,") ",count];

        ];
        Print["***** stopping criteria satisfied at ",
              "(",x,",",y,")", " in ",count," iterations"]
    ;]
```

Example of function call and the output generated:

findRoot[f,0.5,1,0.000000001]

0.5 , 1

next iterate is (0.496644,0.246644) 1

next iterate is (0.998869,0.74551) 2

next iterate is (0.998891,0.997783) 3

next iterate is (1.,0.999999) 4

next iterate is (1.,1.) 5

***** stopping criteria satisfied at (1.,1.) in 5 iterations

The next code defines a function named *gradsearch* which uses the steepest descent method to find the minimizer of a multivariable function from a given starting point and will lists the values of the x_1 and x_2 coordinates at each iteration. The stopping criterion for *gradsearch* allows it to continue to iterate until each element of the gradient vector is less than the given tolerance.

```
f[x_,y_]=x*y^2*E^(-x^2-2y^2)
funx[x_,y_]=D[f[x,y],x]
funy[x_,y_]=D[f[x,y],y]
gradsearch[f_,initx_,inity_,eps_]:=
    Module[{x=initx,y=inity,count=0},Print[x," , ",y];
        component1=N[Abs[funx[x,y]]];
        component2=N[Abs[funy[x,y]]];
        Print[component1, component2];
       While[Max[component1,component2]>eps
                && count<1000,

          count=count+1;
          currentx=funx[x,y];
          currenty=funy[x,y];

            x1=x+t*currentx;
             x2=y+t*currenty;

                 newfun=f[x1,x2];
                funtomin=D[newfun,t];

                      change=FindMinimum[newfun,{t,x},
                       MaxIterations->250];

                Transpose[change];
                sec=change[[2,1]];

                 x1new=N[x+N[t*currentx]]/.sec;
                 x2new=N[y+N[t*currenty]]/.sec;

          Print["next iterate is ",
               "(",x1new,",",x2new,") ",count];

                    x=x1new; y=x2new;

                   component1=N[Abs[funx[x,y]]];
                   component2=N[Abs[funy[x,y]]];
          (*Print[component1,component2];*)
];
          Print["******** stopping criteria satisfied at(",
                  x1new,",",x2new,") in ",count, " iterations."]
;]
```

Example of function call and the output generated

```
gradsearch[f,0,1,0.0000000001]
0 , 1
next iterate is (-0.707107,1.) 1
next iterate is (-0.707107,0.707107) 2
next iterate is (-0.707107,0.707107) 3
******** stopping criteria satisfied at(-0.707107,0.707107 ) in 3 iterations.
```

The following code defines a function called "*renitaroot*" that performs Newton's method for a multivariable function with the stopping criterion that each element of the gradient vector be less than a given tolerance.

```
f[x_,y_]=x^4+2y^4-8x*y+2
funx[x_,y_]=D[f[x,y],x]
funy[x_,y_]=D[f[x,y],y]
grad[x_,y_]={{funx[x,y]},{funy[x,y]}}
funxx[x_,y_]=D[funx[x,y],x]
funyy[x_,y_]=D[funy[x,y],y]
funxy[x_,y_]=D[funx[x,y],y]
hess[x_,y_]={{funxx[x,y],funxy[x,y]},
        {funxy[x,y],funyy[x,y]}}
renitaroot[f_,initx_,inity_,eps_]:=
        Module[{x=initx,y=inity,count=0},
                While[Max[Abs[funx[x,y]],Abs[funy[x,y]]]>eps,

                        count=count+1;
                        diffnew=Inverse[hess[x,y]].grad[x,y];
                        x=x-N[diffnew[[1,1]]];
                        y=y-N[diffnew[[2,1]]];
                ];
          d=MatrixForm[Table[InputForm[{x,y}],
             {r,2},{s,2}]];
         ]
```

By adding the following code, the *"renitaroot"* function can be used iteratively over a grid and the point to which each coordinate converged to is stored in an r x s matrix where r is the number of x-coordinates for which the function is evaluated and s is the number of y-coordinates for which the function was evaluated.

```
For[i=1,i<2,i=i+.5,
        For[j=1,j<2,j=j+.5,

                MatrixForm[Table[renitaroot[f,i,j,.001],{x,r},{y,s}]];
        ]
  ]
```

BIBLIOGRAPHY

[1] Anton, Howard and Albert Herr. 1995. Calculus with Analytic Geometry, Fifth Edition, John Wiley & Sons, Inc. New York.

[2] Asaithambi,N.S.. 1995. Numerical Analysis, Theory and Practice. Saunders College Publishing/Harcourt Brace College Publishing, Fort Worth, Texas.

[3] Boyland,Philip. 1991. Guide to Standard Mathematica Packages. Wolfram Research,Inc., Champaign, llinois.

[4] Bradley, Stephen P.,Arnoldo C. Hax and Thomas L. Magnanti. 1977. Applied Mathematical Programming. Addison-Wesley, Reading, Massachusetts.

[5] Bronson, Richard. 1982. Theory and Problems of Operations Research.McGraw-Hill,Inc., New York.

[6] Chvátal,Vašak. 1983. Linear Programming. W.H.Freeman & Company, New York.

[7] Dennis, J.E. Jr. and Robert B. Schnabel. 1996. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial and Applied Mathematics. Philadelphia.

[8] Gaylord,Richard J.,Samuel N, Kamin and Paul R. Wellin. 1993. Introduction to Programming with Mathematica. TELOS,The Electronic Library of Science. Springer-Verlag Publishers, Santa Clara, California.

[9] Gill,Philip E., Walter Murray and Margaret H. Wright. 1981. Practical Optimization. Academic Press. London.

[10] Hillier, Fredrick S. and Gerald J. Lieberman. 1995. Introduction To Operations Research, Sixth Edition. McGraw-Hill, Inc., New York.

[11] Nocedal, Jorge. 1992. Theory of algorithms for unconstrained optimization, Acta Numerica 1992. Cambridge University Press, New York.

[12] Pfaffenberger, Roger C., and David A. Walker.1976. Mathematical Programming for Economics and Business. The Iowa State University Press, Ames,Iowa.

[13] Rao, S.S..1984. Optimization theory and applications. Halstead Press, New York.

[14] Walsh, Gordon Raymond. 1975. Methods of Optimization. John Wiley & Sons, London.

[15] Wolfram, Stephen. 1991. Mathematica A System for Doing Mathematics by Computer. Addison-Wesley Publishing Company Inc., Redwood City, California.